

175 PTAS
mi computer
96
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
File Edit Search Format Font Style
Untitled
6 lines/inch
THE OMNI-READER
The OMNI-READER is a revolutionary new Text Reader or Optical Character Reader.
It costs a fraction of the price of your computer system. It reads the printed word and translates it into an electronic signal capable of being recognized and down loaded into most computers and word processors. The need for slow and tedious retyping of already printed data is eliminated.
The quick brown fox can jump from the OMNI-READER into
OMNI-READER
Editorial Delta, S.A.
Con este fascículo
GRATIS el 97

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VIII-Fascículo 96

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 -

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-152-6 (tomo 8)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 278511
Impreso en España-Printed in Spain-Noviembre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.

Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.

Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.

Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Dar vida

Robyn Beeche

Entre los postulados de la robótica y la realidad actual existe aún un profundo abismo... Analicemos el tema

En ningún otro campo de la informática existe un vacío tan grande entre la realidad y la fantasía como el que existe en el campo de la robótica. Los robots que vemos en las películas de ciencia-ficción rodadas en Hollywood caminan, hablan y traman la destrucción de la humanidad. Sin embargo, a escasos kilómetros de «la capital del cine», en la Stanford University, los últimos robots de investigación apenas si pueden deambular a través de una habitación sin darse de bruces contra los muebles.

Por supuesto, los robots desempeñan un papel productivo en las fábricas de automóviles y en la industria en general. Pero tales sistemas, generalmente para aplicar pintura o soldadura por arco, se limitan a repetir secuencias de operaciones preprogramadas sin ninguna variación. Son tan «tontos»

«Starlet» cibernética

Los robots prácticos se consideran un desarrollo moderno, pero Rosa Bosom (R*adio*-O*perated* S*imulated Actress* B*attery* O*r* S*tand-by* O*perated* M*ains*), a quien vemos aquí fotografiada junto a su inventor, Bruce Lacey, fue construida en 1966 para hacer el papel de reina de Francia en una producción de *Los tres mosqueteros* en el Royal Court Theatre de Londres. Construida con relés y motores excedentes del gobierno, Rosa acepta instrucciones en forma de tonos musicales, que se convierten en movimiento, y también está equipada con sensores ultrasónicos para detectar obstáculos y evitarlos. En su carrera de actriz, Rosa remedaba sus palabras con voz grabada en cinta, mientras sus labios se movían por control remoto. Lo más interesante, desde el punto de vista cibernético, es que Rosa puede interactuar con un segundo robot llamado Mate. Rosa ha aparecido en varias exhibiciones, incluyendo la Cybernetics Serendipity en el ICA, y más recientemente ganó en Londres el concurso Alternative Miss World (Miss Mundo alternativa)

que, de producirse un vacío en la cinta transportadora que les alcanza el trabajo, arrojarían pintura a ciegas o soldarían en el aire.

Las razones de esta abismal disparidad entre PR y rendimiento son complejas, pero se pueden resumir en dos palabras: *percepción* y *planificación*. En primer lugar, los robots de hoy en día no sacan ningún «sentido» de sus sensores. En segundo lugar, aunque lo hicieran, no sabrían qué hacer. Ésta es la razón por la cual la robótica es un campo de pruebas tan idóneo para aplicar las teorías de inteligencia artificial (AI).

La robótica toca todas las facetas de la AI porque su objetivo es producir un artefacto que pueda enfrentarse con el mundo real. Por consiguiente, un robot ha de extraer algún significado de su entorno.

No es demasiado difícil dotar a un ordenador de toda una variedad de dispositivos de entrada (cámaras de televisión, sensores de calor, exploradores ultrasónicos, teclados de presión, etc.) que puedan darle acceso a una información que los seres humanos no podemos obtener directamente, como la luz infrarroja o la ultravioleta. Pero a menos que el robot habite en un entorno severamente controlado, no comprenderá el significado de sus sensores. La percepción humana, como hemos visto en el capítulo sobre la visión por ordenador, es un proceso permanente de modelación psicológica que se basa en lo que está sucediendo en el mundo real.

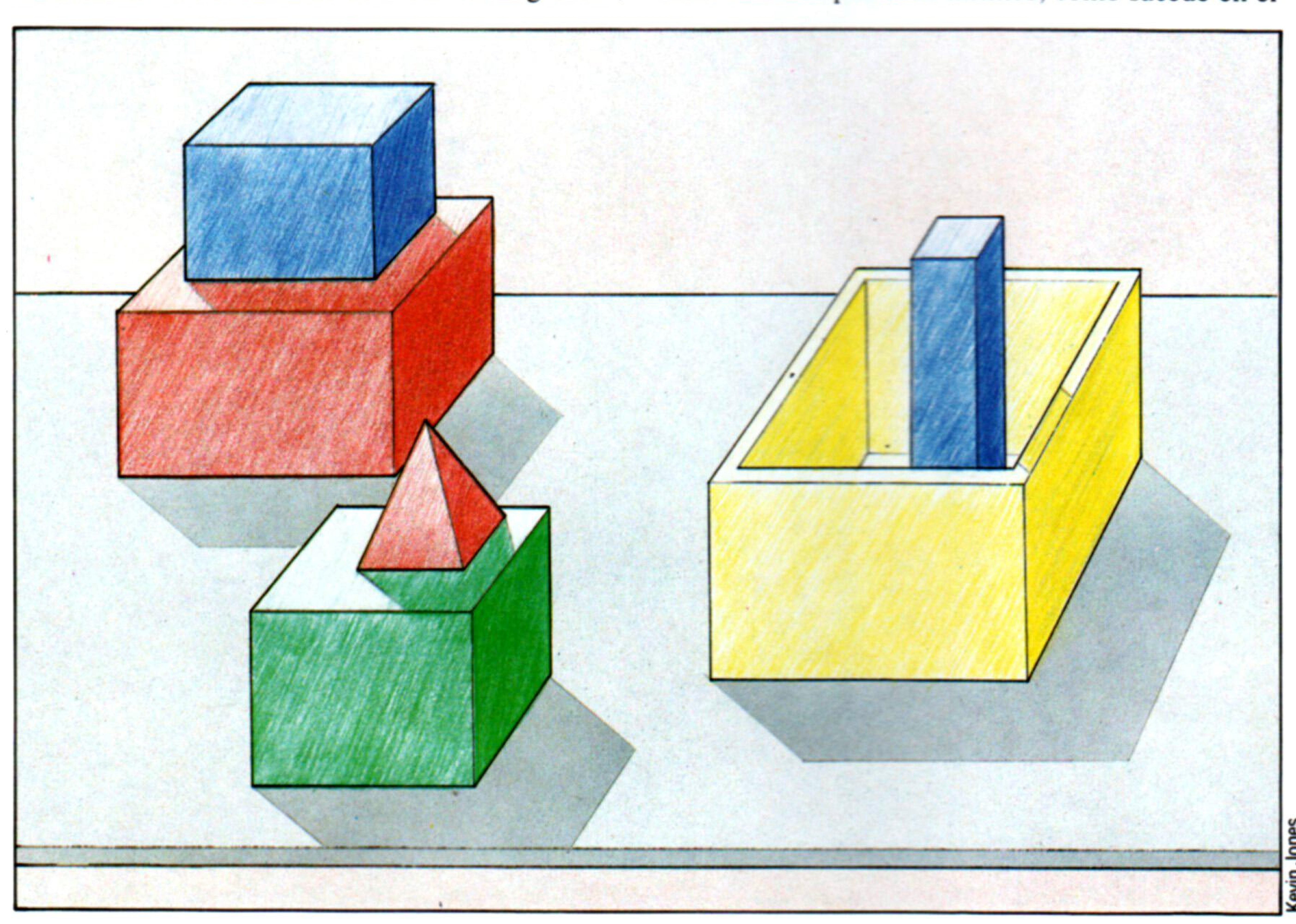
Una cosa es idear un algoritmo eficaz para hallar una ruta a través de un laberinto retenido en la memoria de un ordenador y visualizado en una pantalla, y otra cosa muy distinta utilizar ese algoritmo para conducir un robot a través de un entorno urbano, trozos de setos demasiado crecidos u otros obstáculos insospechados a lo largo del camino.

En el mundo real, la ley suprema es la de Murphy. A ello se debe la enorme popularidad de las competiciones de Micromouse. En estas competiciones, un «ratón» controlado por ordenador ha de navegar hasta el centro de un complejo laberinto de mesa. Desde hace ya muchos años se conocen métodos para hacerlo, al menos en teoría. En la práctica, sin embargo, las paredes nunca son totalmente rectas, quedan manchas resbaladizas allí donde derramaron aceite los contendientes anteriores, etc. En tales circunstancias, la robustez y la adaptabilidad importan más que la elegancia algorítmica.

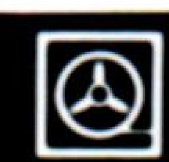
En la mayoría de los otros campos de la AI, los programadores pueden refugiarse en «microcosmos» creados por ellos mismos, como sucede en el caso de un programa para jugar al ajedrez, que opera en un entorno abstracto. Un sistema experto se ocupa de hechos, no de cosas. Esto explica por qué a quienes investigan en el campo de la AI les atrae tanto el llamado *mundo de bloques*.

El *mundo de bloques* es un entorno esquemático simplificado que contiene bloques de construcción de colores de carácter infantil. No son auténticos bloques de construcción, por supuesto, sino *representaciones* formalizadas de esos bloques. Es el lugar predilecto de los científicos dedicados al desarrollo de la AI, cuyos programas pueden resolver problemas y manipular objetos dentro de ese mundo sin renunciar a la comodidad de sus CPU.

Los investigadores en robótica no cuentan con ningún refugio de este tipo. Los robots deben enfrentarse a las vicisitudes de la realidad lo mejor que puedan. Un robot inteligente (uno que se pueda desplazar por su propia iniciativa) debe ser capaz de construir un modelo cognoscitivo de su entorno. Además (para satisfacer décadas de literatura y cine de anticipación) ha de responder a instrucciones habladas. Y puesto que sus respuestas


Kevin Jones

No es el mundo real
El *mundo de bloques* es un modelo sumamente abstracto del mundo real, que permite a los científicos dedicados a la investigación en el campo de la inteligencia artificial evaluar diferentes métodos de análisis de escena, comprensión de textos y solución de problemas. Sin embargo, puede ser difícil traducir con éxito los métodos del mundo de bloques al mundo real



no se pueden prever, sería muy deseable que tuviera cierto grado de capacidad de aprendizaje. Éste es el motivo por el cual realmente no son robots *inteligentes*. Antes de que se pueda construir uno, se deben resolver estos problemas de AI:

- Visión por ordenador.
- Comprensión del habla.
- Resolución de problemas en un entorno dinámico.
- Aprendizaje de la máquina.

y ésta es la lista mínima.

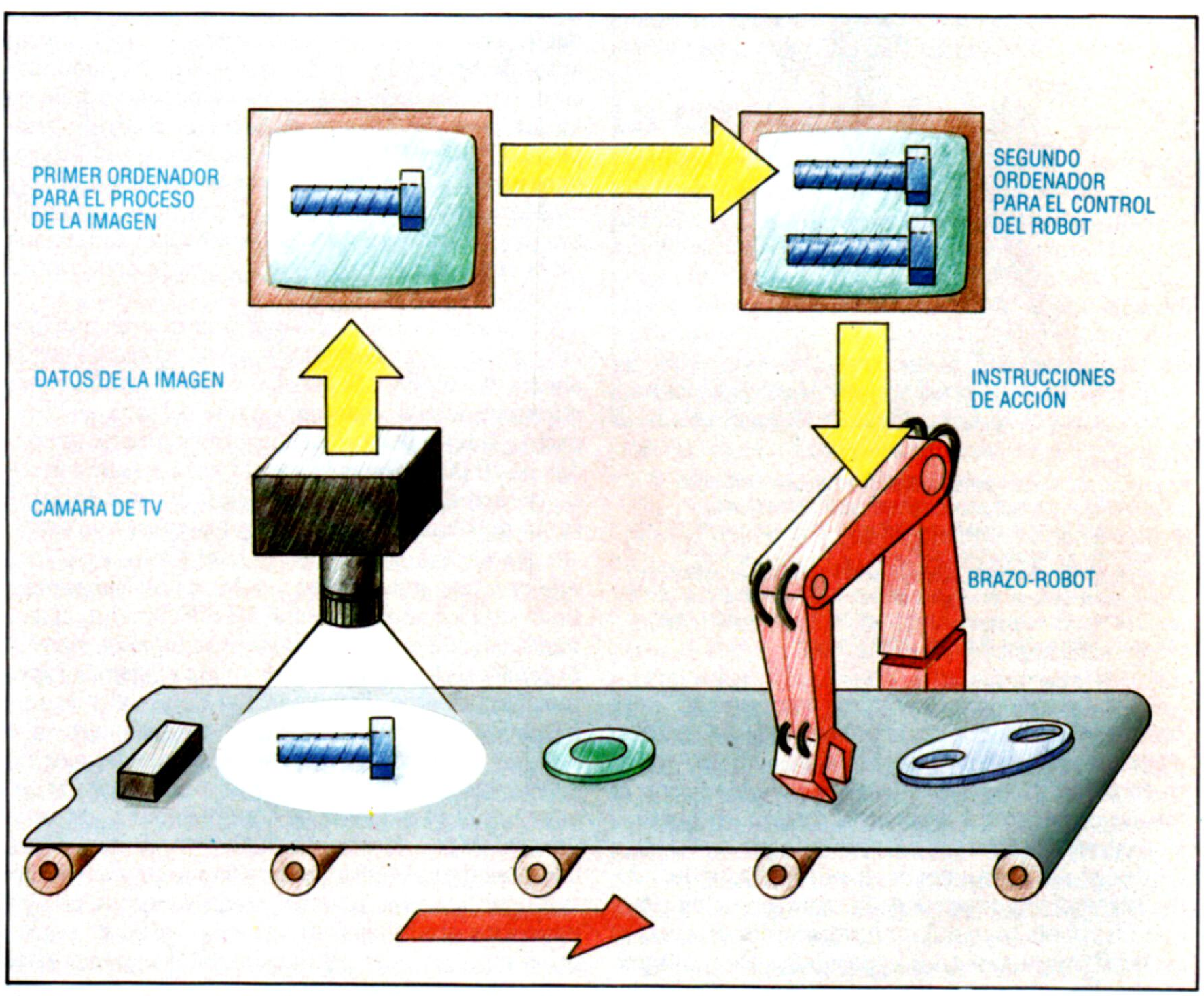
Los robots industriales modernos tienden a diseñarse para entornos sumamente estructurados. Por lo general, un robot no es más que un dispositivo similar a un brazo bajo el control de un ordenador, integrado en una factoría automatizada o en un banco de taller. Su «mano» puede tener atornilladas herramientas y un operador humano puede «enseñarle» una serie de movimientos que el robot puede repetir cuantas veces se precise. Puede memorizar tales secuencias de acciones, pero no modificarlas. Con frecuencia opera en un ambiente regulado tan estrechamente que no necesita en absoluto capacidades perceptuales.

Algunos robots, sin embargo, han dado un pequeño paso hacia el mundo real. Son un poco menos dependientes de las personas u otras máquinas que les proporcionen su trabajo. Pueden, hasta cierto punto, ir a buscarlo ellos mismos.

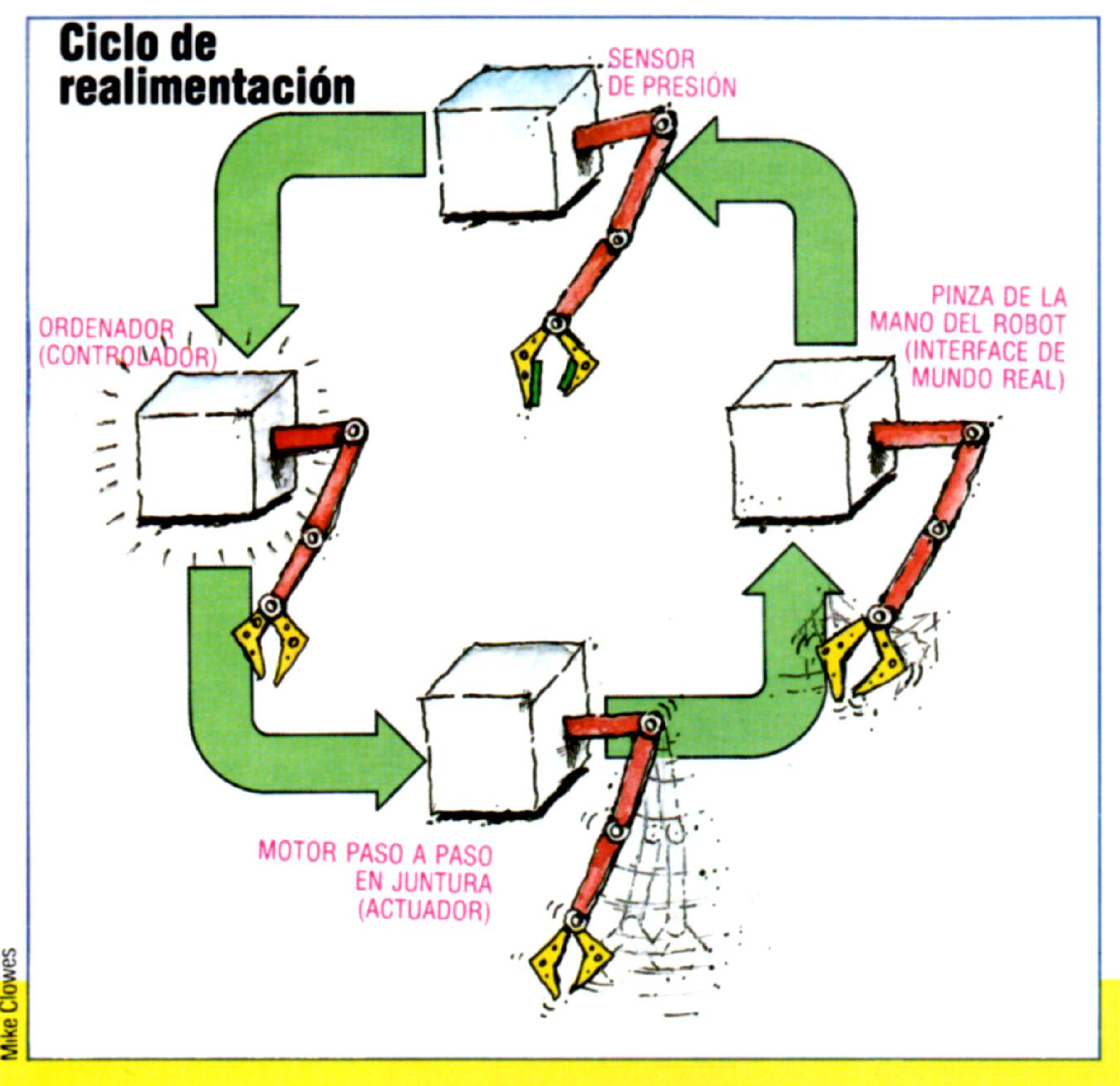
La forma más común de dotar a un robot de un sentido es a través de la vista. Una cámara de video toma imágenes (de un objeto que el robot haya de recoger, p. ej.) y pasa la información de imagen al ordenador de control (éste puede ser un ordenador separado del que controla el brazo del robot). El ordenador procesa la información visual proporcionada (mediante el empleo de métodos de arriba abajo o de abajo arriba, o tal vez de una combinación de ambos) y detecta ciertos patrones que son importantes en el contexto de la tarea del autómata informatizado. Esta información se envía al ordenador que controla el brazo, donde actualiza una descripción simbólica de su entorno ya almacenada allí. El robot posee ahora una imagen de los objetos y las herramientas con las que está trabajando, e instrucciones sobre lo que ha de hacer.

La gran ventaja de un robot dotado de visión, por limitada que ésta sea, es que al autómata informatizado se le pueden proporcionar componentes en diversas orientaciones, y posiblemente también componentes de formas diferentes. Está en condiciones de reconocer el tipo de componente del que se esté ocupando y el ángulo en el cual ha llegado, de modo que podrá manipularlo de modo correcto. Un robot «ciego» por completo intentaría coger cada elemento de la misma manera, probablemente con resultados desastrosos.

Otra forma de hacer que los robots sean más «sensibles» es a través del tacto. Los ingenieros pueden instalar sensores de fuerza o de presión en el extremo de la mano del robot. Cuando éstos entran en contacto con algún objeto, la información se retransmite al ordenador controlador. Si el robot se topa con una pared, el sensor de fuerza volverá a enviarle información de modo que pueda modificar la trayectoria de su brazo para evitar que se repita el accidente.



Transportando la idea
Un robot con capacidad de visión puede utilizarse para reconocer y recoger ciertos elementos de una cinta transportadora. En este caso el autómata informatizado necesitará dos ordenadores: uno para interpretar los datos provenientes de la cámara de televisión, y otro para controlar el mecanismo de prensión



Dedos ágiles
Para manipular objetos con precisión se necesita el sentido del tacto. Siempre que recogemos objetos, estimamos el peso del objeto con el fin de determinar la fuerza de agarre que se requiere para poder levantarlo con éxito. El proceso, que se ha reproducido con niveles de éxito variables en las aplicaciones de robots, esencialmente es un ciclo de realimentación en el cual sensores táctiles vuelven a alimentar los datos al ordenador, que a su vez aprieta o afloja el agarre sobre el objeto

La exhibición que se realiza comúnmente para promocionar los robots sensibles al tacto consiste en hacer que primero perforen un agujero en un trozo de acero fresado y después hacerle realizar lo mismo pero con un huevo, sin romper su cáscara.

Tanto la visión como el tacto, aunque en estadios de desarrollo primitivos en los robots actuales, demuestran la importancia de la realimentación. Para que los robots sean útiles en situaciones diversas es esencial proporcionarles capacidades de realimentación. Por supuesto, es muy difícil transformar información sobre el mundo exterior a una forma que el ordenador del robot pueda utilizar para guiar sus acciones. Uno de los muchos problemas es el de transformar los datos a «tiempo real» (con suficiente rapidez para que el robot responda a lo que está sucediendo antes de que el evento percibido haya terminado). Se dice que el escaso número de robots que están equipados con dispositivos sensoriales y alguna capacidad para interpretar la realimentación son robots de la *segunda generación*.

Ya hemos visto que la comprensión del habla continua era el más arduo de los cuatro tipos de problemas del proceso del lenguaje. La principal dificultad estriba en que el ordenador no posee control alguno sobre lo que se le está diciendo.

No obstante, se puede otorgar al robot un grado útil de reconocimiento de voz (en contraposición a la comprensión del habla). Esto significa tan sólo que se puede entrenar al robot para que responda de la forma adecuada a una pequeña cantidad de palabras y frases. Su comprensión de lo que se le dice es menor que la de un perro que sepa sentarse, tumbarse o ir a buscar algún objeto, pero en una factoría en la cual los obreros humanos pueden tener sus manos ocupadas, incluso estas capacidades lingüísticas tan simples pueden ser de utilidad.

Algunos de los desarrollos más interesantes que se han producido en el campo de la robótica han sido consecuencia de la exploración espacial. Ingenieros de la NASA han diseñado un Mars Rover, un vehículo robótico que puede rodar sobre la superficie del planeta Marte. El vehículo dispone de sensores, como cámaras de televisión, para recoger información sobre sus inmediaciones, y la inteligencia suficiente para hacer uso de la misma. No le es dable esperar señales provenientes de la Tierra antes de decidir lo que tiene que hacer a continuación, pues los mensajes enviados por radio pueden tardar hasta media hora en hacer el recorrido completo. El Mars Rover está planeado para el futuro, pero en 1976 los norteamericanos hicieron aterrizar en Marte dos sondas Viking. Éstas pudieron analizar muestras del suelo y de la atmósfera marciana sin la supervisión paso a paso por parte del «control de misión».

Colocar robots en el espacio para que trabajen en lugar de seres humanos supone un considerable ahorro de dinero. Según las últimas estimaciones norteamericanas, mantener a una persona en el espacio y traerla de vuelta sana y salva ¡cuesta alrededor de 10 000 dólares la hora! Quizá resulte oneroso desarrollar robots espaciales, pero a la larga serán más baratos que sus equivalentes humanos.

Una posibilidad exótica para el futuro es la idea que presentó por primera vez John von Neumann a finales de los años cuarenta. Von Neumann, el matemático que sentó los cimientos teóricos para el ordenador digital electrónico, hizo además otros muchos trabajos, incluyendo el desarrollo de una teoría de autómatas capaces de reproducirse a sí mismos. Él propuso que las máquinas podían reproducirse siguiendo un sencillo conjunto de reglas. Según el esquema de Von Neumann, un sistema robótico autorreproductivo necesita cuatro componentes. El primero es una factoría automatizada que reúna la materia prima y la convierta en productos según las instrucciones dadas. El segundo componente es un duplicador que copie estas instrucciones. El tercero es un controlador que pase las instrucciones al duplicador para que las copie. El cuarto y último componente es el conjunto de las propias instrucciones, que le dice al sistema cómo construir una nueva factoría automática completa a partir de los productos que haya fabricado.

Durante casi 40 años éstos fueron apenas conceptos teóricos, hasta que los investigadores de la NASA propusieron un anteproyecto de ingeniería para un sistema que se duplicara a sí mismo en la Luna. Tal factoría espacial contendría un «constructor universal» que tomaría los componentes fabricados por la unidad de producción y construiría con ellos una nueva factoría, incluyendo, por supuesto, otro constructor universal. Se valdría de las materias primas de la Luna y no requeriría servicios de mantenimiento desde la Tierra. Asimismo, podría proveerse a sí mismo de un buen contingente de nuevos obreros robot.

Puesta a punto

Añadiremos algunas rutinas para manipular la pila, además de las útiles implementaciones de la recursión

Cuando se escribe un programa de este tipo, por lo general se necesitan algunas rutinas de utilidad general.

Entre éstas se incluyen procedimientos para actualizar el tablero, comprobar la legalidad de los movimientos, etc. Por ejemplo, en un programa para jugar al ajedrez sería muy útil una rutina para comprobar si el rey está o no en jaque.

Por supuesto, podríamos escribir una rutina para hallar movimientos que pongan en jaque al rey del adversario y, durante la defensa, posiciones que mantengan a nuestro rey a salvo de tal eventualidad. Sin embargo, tiene mucho más sentido contar con una rutina general que asigne un valor TRUE o FALSE a "¿está el rey de COLor en jaque?". Luego podemos llamar a la rutina con diferentes valores de COLor y decidir después qué hacer con el resultado.

Para nuestro programa de *go* vamos a adoptar un enfoque similar. En lugar de tener rutinas específicas que decidan qué hacer con diversos tipos de grupos, implementaremos una rutina general para contar los miembros de cualquier grupo determinado. Entonces podremos decidir, en las rutinas de llamada separadas, qué significado tienen los resultados.

El procedimiento principal para hacer esto se denomina PROCbuscar en la versión para el BBC Micro. Se le pasan dos parámetros: la posición de todas las fichas del grupo que deseemos contar (P%) y el color del grupo de fichas (C%). Luego el procedimiento le suma 1 a CSTN%, que se utiliza para totalizar la cantidad de fichas del grupo, y observa las cuatro fichas adyacentes. Para cada una de estas fichas adyacentes, lleva a cabo el mismo procedimiento, pero sólo si la ficha no se ha contado todavía. Por supuesto, ya disponemos de un procedimiento para llevar a cabo esta búsqueda en cuatro direcciones desde una ficha dada, ¡de modo que la forma obvia de contar las fichas adyacentes consiste en que el procedimiento se llame a sí mismo! Éste es un ejemplo de recursión y puede que a primera vista parezca complicado. Para esclarecer un poco las cosas, he aquí un análisis estructurado de la rutina de búsqueda:

```
BUSCAR (desde-posición,para-color)
 IF desde-posición se halla en el tablero
  AND desde-posición contiene una ficha de para-color
  AND todavía no hemos contado desde-posición
 THEN
  marcar desde-posición, para que no volvamos a
  contarla
  sumar uno a CSTN% (nuestra variable del contador)
  BUSCAR (norte de desde-posición, para-color)
  BUSCAR (este de desde-posición, para-color)
  BUSCAR (sur de desde-posición, para-color)
  BUSCAR (oeste de desde-posición, para-color)
 ENDIF
END BUSCAR
```

Observe que las llamadas recursivas se tratan como cuatro sentencias separadas, y no en un bucle, para evitar problemas cuando la recursión es profunda.

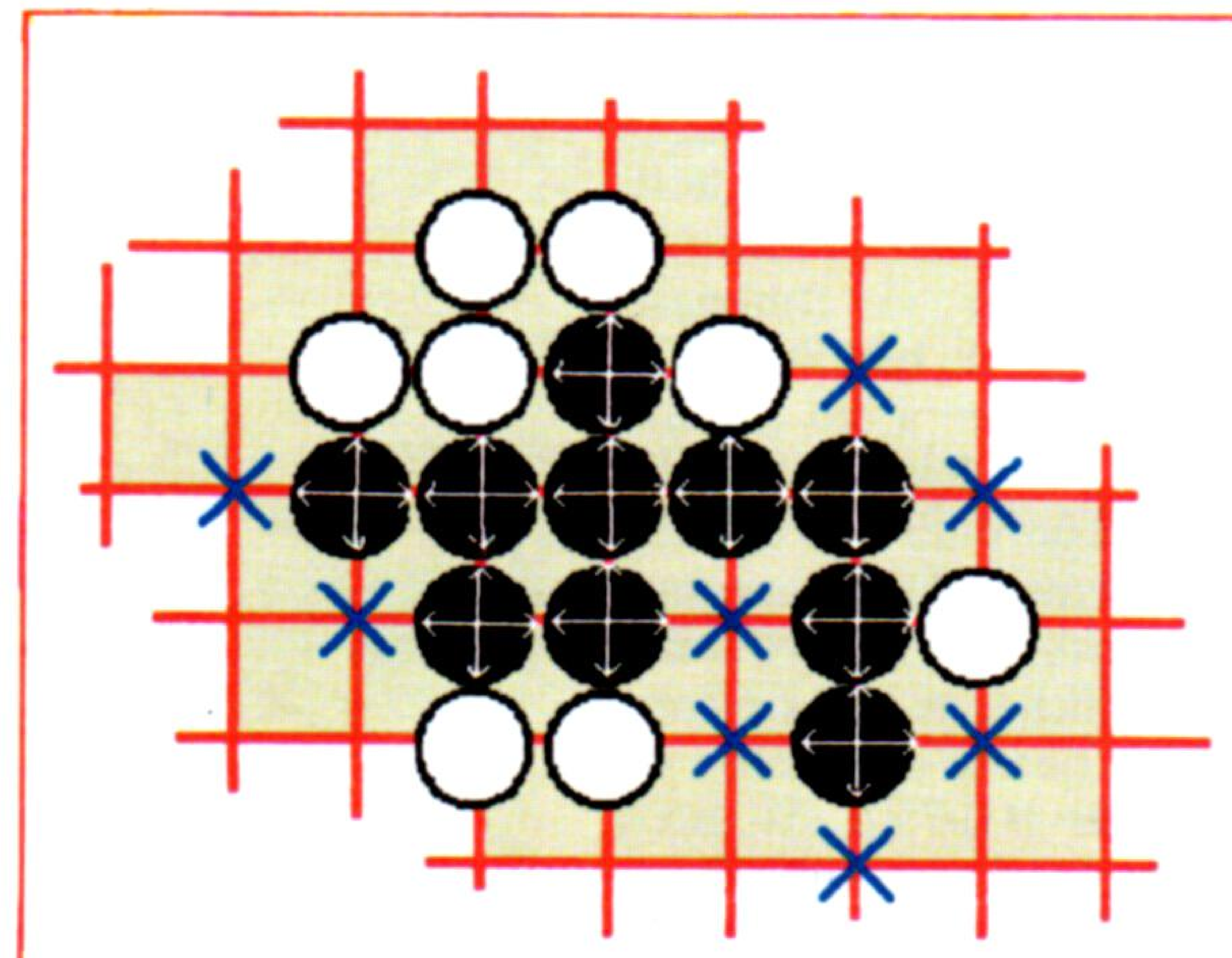
La estructura de pila

Cuando la rutina se llama a sí misma, el parámetro *desde-posición* se reemplaza por la nueva *desde-posición* (en una de las cuatro direcciones) y la rutina prosigue. La nueva *desde-posición* también puede llamar de forma recursiva a la rutina de búsqueda, creando otra variable *desde-posición,* y así sucesivamente. En algún punto, una de las condiciones IF será falsa, de modo que la llamada en cuestión terminará sin que las sentencias THEN realicen una llamada recursiva. Siempre que la recursión retorna de un nivel como éste, se restablece el valor anterior de *desde-posición*.

Se puede pensar en esta estructura de datos de «pila» como en una pila de platos. Cuando la rutina es llamada recursivamente, por arriba de la pila se añade un nuevo plato. Sin embargo, cuando acabe este nivel particular de la recursión, este mismo plato se debe quitar de la pila (restableciéndose el mismo valor de variable). Esta estructura se denomina LIFO (*Last In, First Out*: último en entrar, primero en salir).

Lamentablemente, aunque el BASIC BBC admite la recursión en el pase de parámetros de PROCedimientos y variables localizadas, el Commodore 64, Sinclair Spectrum y Amstrad CPC 464/664 carecen de esta facilidad. Por consiguiente, en estas rutinas usted encontrará las matrices llamadas SK%() o s(), que "apilan" los valores necesarios.

Ello se realiza reteniendo un puntero (PILA% o su equivalente) que llevan un registro de la posición



Contando cabezas

La rutina de búsqueda, que se utiliza para evaluar el estado de los grupos del tablero, incorpora (o simula) procedimientos recursivos para buscar en las cuatro direcciones desde cada una de las fichas del grupo. La rutina termina cuando se llega a una licencia o a una ficha del enemigo

superior de la pila. Cada vez que se coloque algo en la pila, se incrementa este puntero, y se lo reduce cada vez que se saca algo. De esta manera, la variable adecuada se restablece al valor de la parte superior de la pila.

Un efecto secundario de esta rutina de búsqueda es que la recursión siempre termina en los márgenes del grupo que esté examinando. Cuando buscamos estos márgenes, podemos comprobar si se hallan en el margen del tablero o si contienen una ficha del color contrario. Si ninguna de estas condiciones es verdadera, necesariamente ha de ser una licencia del grupo. Por consiguiente, la segunda parte de esta rutina cuenta la cantidad de licencias relacionadas con el grupo, utilizando la variable CLIB%. Nuevamente, estas posiciones se han de marcar tras haberlas contado, ya que de lo contrario se correría el riesgo de contarlas dos veces.

Al contar un grupo de fichas, siempre llamaremos a la rutina PROCcontar, que a su vez llama a PROCbuscar. Ésta inicializará las variables del contador y limpiará los marcadores tras utilizar la rutina de búsqueda.

Al marcar posiciones, observará que empleamos el propio tablero real, para evitar las cuentas dobles. No obstante, tras haber marcado el tablero, hemos de poder suprimir los marcadores antes de continuar. Nuevamente hemos utilizado una rutina general (PROClimpiar), que opera mediante AND los bytes del tablero apropiados con 3, que suprime el marcador del bit 2. Puesto que los dos bits menos significativos de cada byte del tablero contienen los colores, y que los marcadores de ficha y licencia están retenidos en el tercero y el cuarto bit, la llamada PROClimpiar(3) es ideal.

Por ejemplo:

Byte:	A	B	C	D	E	F	G	H
Máscara:	0	0	0	0	0	0	1	1
Resultado:	0	0	0	0	0	0	G	H

La versión para el Spectrum tiene un aspecto diferente, porque en el Spectrum la instrucción AND no desenmascara los bits en los que no estamos interesados. Sólo se puede utilizar para dar resultados verdadero/falso en expresiones tales como: IF X<3 AND Y=3 THEN... En consecuencia, hemos de recurrir a una operación de resta aritmética normal para limpiar el bit 2 en la versión de esta rutina para el Spectrum.

Definiendo esta rutina de limpieza general, también podemos utilizarla para limpiar el tablero antes de una partida, llamando a la rutina con parámetro cero. Esto se ha añadido en la línea 1400.

Sólo nos resta un procedimiento del juego antes de que podamos comenzar a escribir las verdaderas rutinas para procesar los movimientos. Se trata del código que organiza el tablero al comienzo del juego con las adecuadas fichas de handicap. Tal como explicáramos en el primer capítulo de esta serie, el *go* posee un método exclusivo para equilibrar los niveles de destreza, al permitir que el jugador más débil (quien juega siempre con las negras) coloque entre dos y nueve fichas en el tablero como su primer movimiento. Estas fichas se han de colocar en posiciones específicas, que se establecen en la rutina leer handicaps (líneas 600-750). Si usted ejecutó la primera parte del programa que ofrecimos anteriormente, habrá observado que se le indicó que entrara un número de handicap al final de la rutina pantalla_titulos (pero entonces ese número era ignorado). Añadiendo la línea 1580 y la rutina desde la línea 1630 a la 1690, ahora las fichas de handicap se colocarán correctamente; las sentencias DATA entre las líneas 670 y 740 dan las posicio-

Módulo dos

BBC Micro

```
270 PROCleer_handicaps
590 :
600 DEF PROCleer_handicaps
610 LOCAL L%
620 DIM hncp% 43
630 RESTORE 670
640 FOR L% = 0 TO 43
650    READ hncp%?L%
660    NEXT
670 DATA &44,&CC
680 DATA &44,&CC,&4C
690 DATA &44,&CC,&4C,&C4
700 DATA &44,&CC,&4C,&C4,&88
710 DATA &44.&CC.&4C,&C4,&84,&8C
720 DATA &44,&CC,&4C,&C4,&84,&8C,
    &88
730 DATA &44,&CC,&4C,&C4,&84,&8C,
    &48,&C8
740 DATA &44,&CC,&4C,&C4,&84,&8C.
    &48,&C8,&88
750 ENDPROC
760 :
770 REM ****************************************
1400 PROClimpiar(0)
1580 PROChandicap(hand%)
1620 :
1630 DEF PROChandicap(hand%)
1640 LOCAL L%,P%,Q%
1650 Q%=INT((hand%-2)/2*(hand%+1)+0.5)
1660 FOR L%=Q% TO Q%+hand%-1
1670    P%=hncp%?L% : tablero%?P%=negras%
1680    NEXT
1690 ENDPROC
1700 :
1710 REM ****************************************
4030 :
4040 DEF PROCcontar(P%,C%)
4050 clib%=0 : cstn%=0
4070 PROCbuscar(P%,C%)
4080 PROClimpiar(color%)
4090 ENDPROC
4100 :
4110 REM ****************************************
4120 :
4130 DEF PROCbuscar(P%,C%)
4140 IF (P% AND 240)=0 OR (P% AND 15)=0 THEN
    ENDPROC
4150 IF (tablero%?P% AND color%)=0 THEN
    4250
4160 IF (tablero%?P% AND C%)=0 THEN
    ENDPROC
4170 IF (tablero%?P% AND marcador%)>0 THEN
    ENDPROC
4180 tablero%?P%=C%+marcador%
4190 cstn%=cstn%+1
4200 PROCbuscar(P%+dir%(1),C%)
4210 PROCbuscar(P%+dir%(2),C%)
4220 PROCbuscar(P%+dir%(3),C%)
4230 PROCbuscar(P%+dir%(4),C%)
4240 ENDPROC
4250 IF (tablero%?P% AND licencia%)>0 THEN
    ENDPROC
4260 tablero%?P%=licencia%
4270 clib%=clib%+1
4290 ENDPROC
4300 :
4310 REM ****************************************
4320 :
4330 DEF PROClimpiar(M%)
4340 LOCAL L%
4350 FOR L%=0 TO 255
4360    tablero%?L%=tablero%?L% AND M%
4370    NEXT
4380 ENDPROC
4390 :
4400 REM ****************************************
```

nes correctas del tablero, correspondiendo cada sentencia a un número de fichas de handicap entre dos y siete.

En el próximo capítulo añadiremos las rutinas que se requieren para jugar de la manera apropiada. Debido a la forma estructurada en que se ha desarrollado el programa, también podremos modificarlo fácilmente para permitir partidas entre dos jugadores, con el ordenador vigilando que nadie haga trampa.

Commodore 64

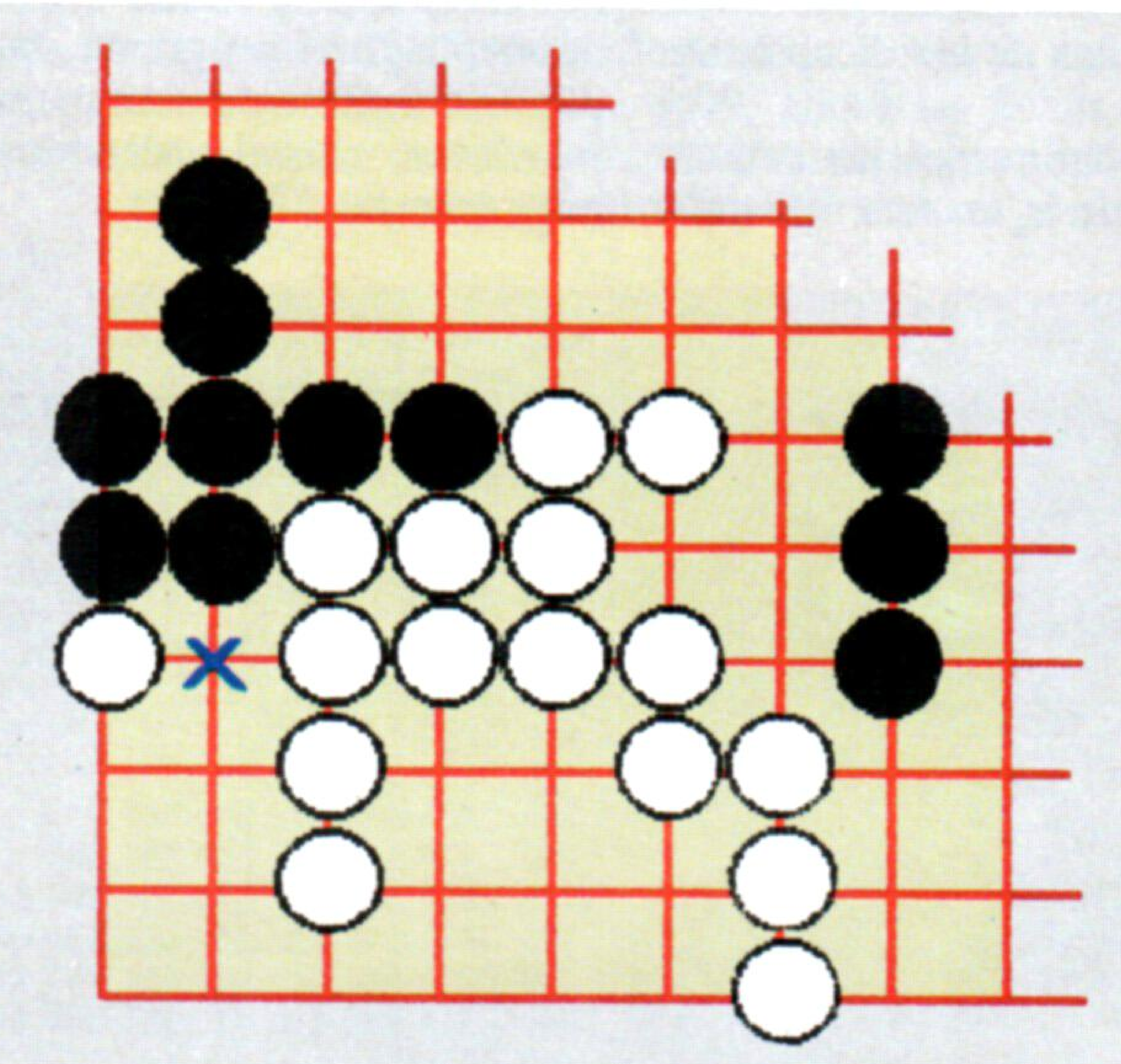
```
320 GOSUB 600
590 :
600 REM RUTINA LEER-HANDICAPS
620 HNCP=TABLERO+512
640 FOR L=0 TO 43
650 READ H%:POKE HNCP+L,H%
660 NEXT
670 DATA 68,204
680 DATA 68,204,76
690 DATA 68,204,76,196
700 DATA 68,204,76,196,136
710 DATA 68,204,76,196,132,140
720 DATA 68,204,76,196,132,140,136
730 DATA 68,204,76,196,132,140,72,200
740 DATA 68,204,76,196,132,140,72,200,136
750 RETURN
760 :
770 REM ****************************************
1400 MSK%=0:GOSUB 4330
1580 GOSUB 1630
1620 :
1630 REM RUTINA HANDICAP
1650 Q%=INT((HND%-2)/2*(HND%+1)+0.5)
1660 FOR L=Q% TO Q%+HND%-1
1670 P%=PEEK(HNCP+L):POKE TABLERO+P%,
     NEGRAS%
1680 NEXT
1690 RETURN
1700 :
1710 REM ****************************************
4030 :
4040 REM RUTINA CONTADOR
4050 CLIB%=0:CSTN%=0
4070 SP%=CP%:SC%=CC%:GOSUB 4130
4080 MSK%=COLOR%:GOSUB 4330
4090 RETURN 0 THEN RETURN
4095 IF (PEEK(TABLERO + SP%) AND COLOR%)=0 GOTO
     4250
4096 IF (PEEK(TABLERO + SP%) AND SC%)=0 THEN
     RETURN
4097 IF (PEEK(TABLERO + SP%) AND MARCADOR%) > 0
     THEN RETURN
4100 :
4110 REM ****************************************
4120 :
4130 REM RUTINA BUSQUEDA
4140 IF (SP% AND 240)=0 OR (SP% AND 15)=0
4180 POKE TABLERO + SP%,SC% + MARCADOR%
4190 CSTN=CSTN%+1
4195 SK%(PILA%)=SP%:PILA%=PILA%+1
4200 SP%=SK%(PILA%-1)+DIR%(1):GOSUB
     4130
4210 SP%=SK%(PILA%-1)+DIR%(2):GOSUB
     4130
4220 SP%=SK%(PILA%-1)+DIR%(3):GOSUB
     4130
4230 SP%=SK%(PILA%-1)+DIR%(4):GOSUB
     4130
4235 PILA%=PILA%-1:SP%=SK%(PILA%)
4240 RETURN
4250 IF (PEEK(TABLERO+SP%) AND LICENCIA%)>0 THEN
     RETURN
4260 POKE TABLERO+SP%,LICENCIA%
4270 CLIB%=CLIB%+1
4290 RETURN
4300 :
4310 REM ****************************************
4320 :
4330 REM RUTINA LIMPIEZA
4350 FOR L=0 TO 255
4360 POKE TABLERO+L,PEEK(TABLERO+L)AND
     MSK%
4370 NEXT
4380 RETURN
4390 :
4400 REM ****************************************
```

Amstrad CPC 464/664

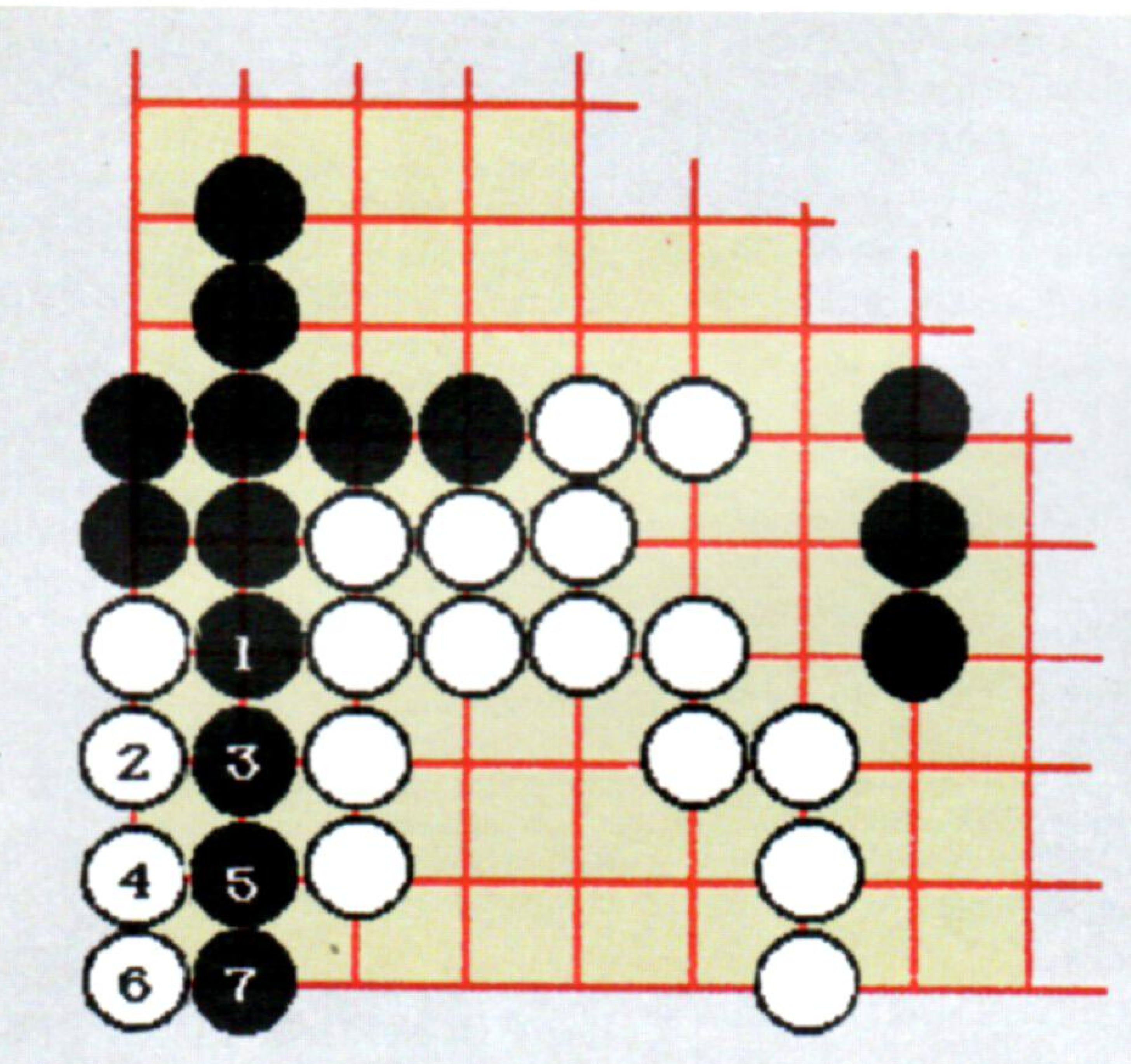
```
270 GOSUB 600:REM leer handicaps
590 :
600 REM rutina leer handicaps
620 hncp=tablero+&200
630 RESTORE 670
640 FOR l%=0 TO 43
650 READ h%:POKE(hncp+l%),h%
660 NEXT l%
670 DATA &44,&cc
680 DATA &44,&cc,&4c
690 DATA &44,&cc,&4c,&c4
700 DATA &44,&cc,&4c,&c4,&88
710 DATA &44,&cc,&4c,&c4,&84,&8c
720 DATA &44,&cc,&4c,&c4,&84,&8c,&88
730 DATA &44,&cc,&4c,&c4,&84,&8c,&48,&c8
740 DATA &44,&cc,&4c,&c4,&84,&8c,&48,&c8,&88
750 RETURN
760 :
770 REM ****************************************
1400 mascara%=0:GOSUB 4330:REM rutina limpieza
1580 GOSUB 1630:REM rutina handicap
1620 :
1630 REM rutina handicap
1650 q%=INT((hand%-2)/2*(hand%+1)+0.5)
1660 FOR l%=q% TO q%+hand%-1
1670 LET p%=PEEK(hncp+l%):POKE(tablero+p%),
     negras%
1680 NEXT l%
1690 RETURN
1700 :
1710 REM ****************************************
4030 :
4040 REM rutina contador
4050 clib%=0:cstn%=0
4060 cloc%(1)=0:cloc%(2)=0
4070 sp%=cp%:sc%=cc%:GOSUB 4130:REM
     busqueda
4080 mascara%=color%:GOSUB 4330: REM
     limpiar
4090 RETURN
4110 REM ****************************************
4120 :
4130 REM rutina de busqueda
4140 IF (sp%AND 240)=0 OR (sp%AND 15)=0 THEN
     RETURN
4150 IF (PEEK(tablero + sp%)AND color%)=0 THEN
     4250
4160 IF (PEEK(tablero + sp%)AND sc%)= THEN
     RETURN
4170 IF (PEEK(tablero + sp%)AND marcador%)>0 THEN
     RETURN
4180 POKE(tablero + sp%),sc% + marcador%
4190 cstn%=cstn% + 1
4195 s(pila%)=sp%:pila%=pila% + 1
4200 sp%=s(pila%-1)+dir%(1):GOSUB 4130
4210 sp%=s(pila%-1)+dir%(2):GOSUB 4130
4220 sp%=s(pila%-1)+dir%(3):GOSUB 4130
4230 sp%=s(pila%-1)+dir%(4):GOSUB 4130
4235 pila%=pila%-1:sp%=s(pila%)
4240 RETURN
4250 IF (PEEK(tablero + sp%)AND licencia%)>0 THEN
     RETURN
4260 POKE(tablero+sp%),licencia%
4270 clib%=clib%+1
4290 RETURN
4300 :
4310 REM ****************************************
4320 :
4330 REM rutina limpieza
4350 FOR l%=0 TO 255
4360 POKE (tablero+l%),(PEEK(tablero+l%) AND
     mascara%)
4370 NEXT l%
4380 RETURN
4390 :
4400 REM ****************************************
```

Sinclair Spectrum

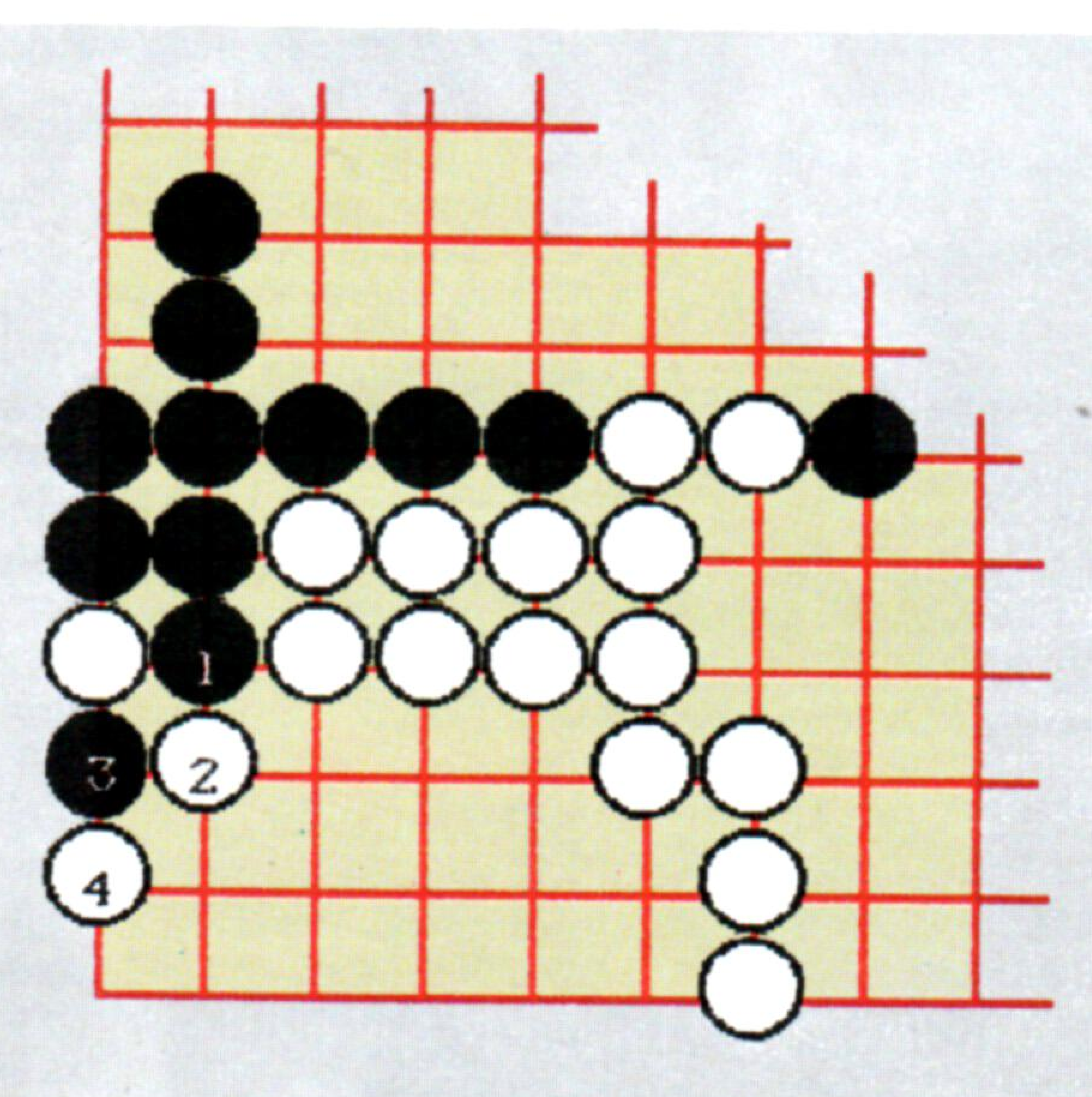
```
 270 GO SUB 600
 590 :
 600 REM rutina leer-handicaps
 620 LET hncp=tablero + 512
 630 RESTORE 670
 640 FOR 1=0 TO 43
 650 READ h:POKE hncp+1,h
 660 NEXT 1
 670 DATA 68,204
 680 DATA 68,204,76
 690 DATA 68,204,76,196
 700 DATA 68,204,76,196,136
 710 DATA 68,204,76,196,132,
     140
 720 DATA 68,204,76,196,132,
     140,136
 730 DATA 68,204,76,196,132,
     140,72,200
 740 DATA 68,204,76,196,132,
     140,72,200,136
 750 RETURN
 760 :
 770 REM *******************************
1400 LET mascara=0:GO SUB 4330
1580 GO SUB 1630
1620 :
1630 REM rutina handicap
1650 LET q=INT((hand-2)/2* (hand+1)
       +0.5)
1660 FOR I=q TO q+hand-1
1670 LET p=PEEK (hncp+1):POKE tablero+
       p,negras
1680 NEXT I
1690 RETURN
1700 :
1710 REM *******************************
4030 :
4040 REM rutina contador
4050 LET clib=0:LET cstn=0
4070 LET sp=cp:LET sc=cc: GO SUB
     4130
4080 LET mascara=color:GO SUB
     4330
4090 RETURN
4100 :
4110 REM *******************************
4120 :
4130 REM rutina de busqueda
4140 IF INT (sp/16)=0 OR sp-16* INT (sp/16)
       =0 THEN RETURN
4150 IF PEEK (tablero+sp)=licencia OR PEEK
       (tablero+sp)=0 THEN GO TO 4250
4160 IF PEEK (tablero + sp)= color-sc THEN
       RETURN
4170 IF PEEK (tablero + sp)> color THEN
       RETURN
4180 POKE tablero + sp, sc + marcador
4190 LET cstn=cstn + 1
4195 LET s(pila)=sp:LET pila=pila + 1
4200 LET sp=s(pila-1)+d(1): GO SUB
       4130
4230 LET sp=s(pila-1)+d(4): GO SUB
       4130
4235 LET pila=pila-1:LET sp=s(pila)
4240 RETURN
4250 IF PEEK (tablero+sp)>color THEN
       RETURN
4260 POKE tablero + sp,licencia
4270 LET clib=clib + 1
4290 RETURN
4300 :
4310 REM *******************************
4320 :
4330 REM rutina limpieza
4350 FOR I=0 TO 255
4360 IF PEEK (tablero + 1)>mascara THEN POKE
       (tablero +), PEEK (tablero +
       1)-mascara-1:
       GO TO 4360
4370 NEXT I
4380 RETURN
4390 :
4400 REM *******************************
```



Ventaja territorial
En esta situación las blancas han conquistado la esquina inferior izquierda del tablero. Sin embargo, este territorio aún no está completamente a salvo. El punto clave es el señalado mediante la cruz



Poniéndose a cubierto
Si se permite a las negras jugar en este punto clave, este movimiento se denominará un *Atari* en la ficha blanca del borde del tablero. Las blancas podrían intentar «correr» hasta la esquina, pero no pueden impedir una ulterior captura por parte de las negras



Maniobra alternativa
En vez de correr hasta la esquina, las blancas podrían sacrificar la ficha situada en el borde del tablero y, de este modo, detener la irrupción de las negras en su territorio jugando como se indica

Caroline Clayton

Para leerlo todo
El Omni-reader, de Oberon, es el primer dispositivo para reconocimiento óptico de caracteres (OCR) a la vez fiable y dentro de la gama de precios para las empresas pequeñas y medianas. Puede leer texto de una página impresa y transferirlo directamente al software de tratamiento de textos de un ordenador a través de una interface en serie RS232C. Esto permite editar el texto en pantalla sin tener que digitar el documento completo. Por el momento, Oberon sólo ha proporcionado software para los micros de gestión más populares, aunque cualquier ordenador con una interface compatible con RS232C adecuada puede utilizar el sistema



Crispin Thomas

Un ojo avizor

Su ordenador, gracias al Omni-reader, podría leer estas mismas palabras que usted está leyendo. Analicemos este interesante dispositivo

En los últimos años se han hecho muchas especulaciones sobre el concepto y el desarrollo de la «oficina sin papeles». La idea proviene del hecho de que, puesto que se pueden transferir datos directamente de un ordenador a otro, en teoría no existe ninguna necesidad de consignar la información en papel. Sin embargo, las predicciones concernientes a esta situación ideal han resultado ser algo prematuras. A la mayoría de nosotros el uso de lápiz y papel sigue resultándonos el método más conveniente para anotar información. Asimismo, durante la fase de «transición» entre la oficina actual y la «oficina sin papeles», los datos aún se deben digitar de forma manual en el ordenador. Para acelerar este proceso, los dispositivos OCR (*optical character recognition:* reconocimiento óptico de caracteres), que aparecieron por primera vez en 1955, son capaces de leer el texto de una página impresa. No obstante, sólo recientemente se han convertido en propuestas prácticas y fiables para su uso en la oficina.

El Omni-reader, de Oberon International, es uno de los nuevos OCR basados en microprocesador que ha aparecido para micros de gestión. Aunque inicialmente el único software disponible es para máquinas tales como el IBM PC y sus compatibles, la gama Apricot y el Apple Macintosh, es factible utilizar el dispositivo con cualquier micro equipado con una interface RS232C adecuada.

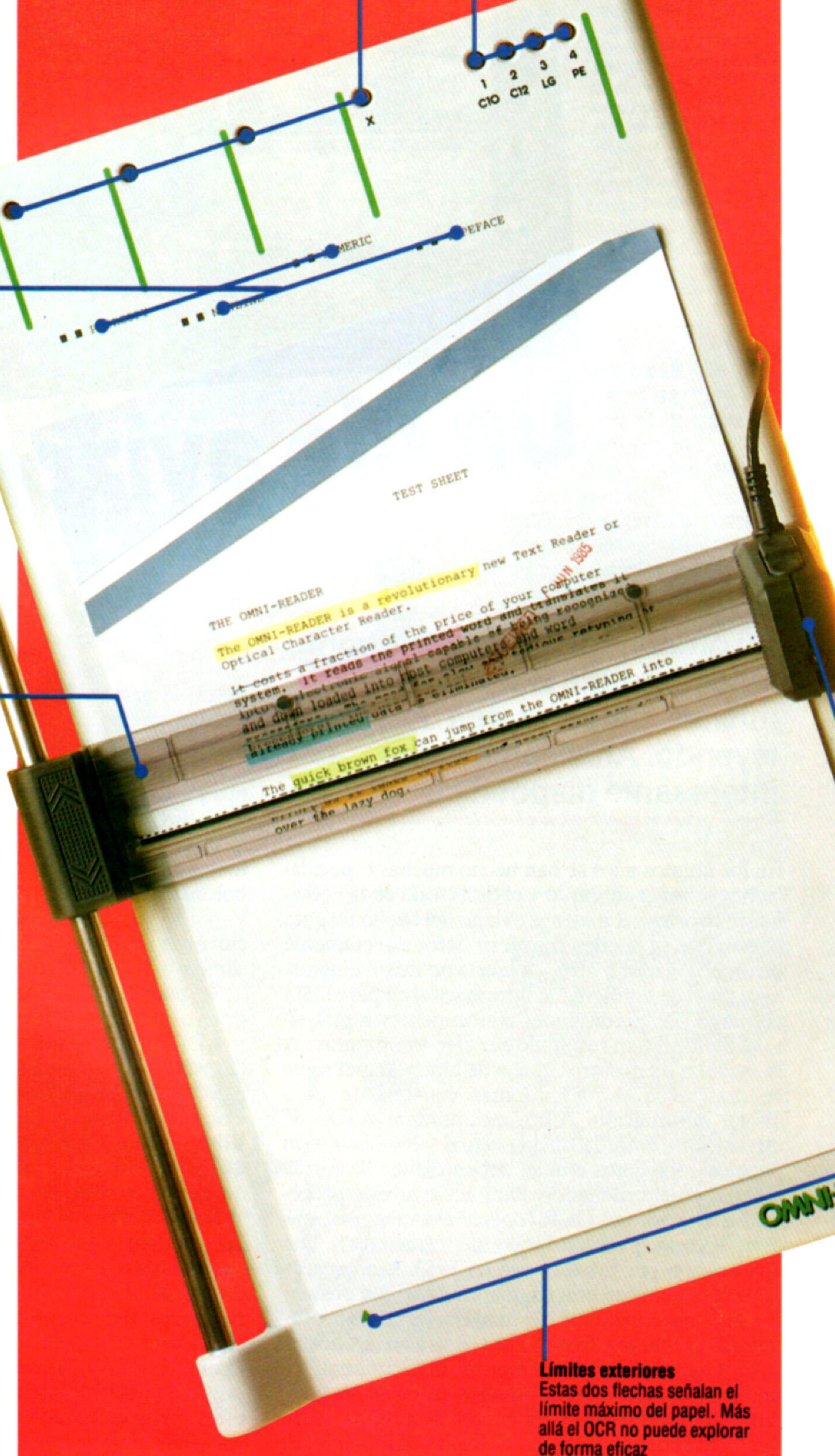
El Omni-reader se compone de una tablilla plástica para documentos con una regla horizontal que se desliza a lo largo de una barra vertical situada a la izquierda de la tablilla. El lector óptico propiamente dicho (un dispositivo manual que tiene dos botones y un LED arriba) está montado en la regla y, por lo tanto, se puede mover a través de un documento colocado sobre la tablilla, de forma muy similar a la de un plotter *xy*.

Se pueden seleccionar numerosas funciones y tipos de letra diferentes, que son indicadas mediante LEDS a lo largo de la parte superior de la tablilla. Actualmente el Omni-reader sólo puede explorar cuatro tipos de letra o «fuentes». Éstos son el Courier 10, Courier 12, Letter Gothic 12 y Prestige Elite 12, que son las cuatro fuentes más populares que se utilizan en las impresoras margarita y las máquinas de escribir electrónicas.

En el interior del lector óptico hay dos fuentes de luz infrarroja a ambos lados de una lente que enfoca la imagen sobre el sistema de circuitos detector de luz. El detector sólo puede leer caracteres impresos con tintas basadas en carbón o toner de fotocopiadoras (que suelen tener una base de carbón), lo que aporta ventajas e inconvenientes. En el lado positivo, significa que las tintas de los bolígrafos y otras tintas no basadas en carbón son «transparentes» para el detector. Por tanto, se puede leer el texto apropiado aun cuando haya sido marcado con

un bolígrafo o un sello de goma. Además, también significa que la mayor parte de los papeles de color no afectarán la operación del Omni-reader. No obstante, las limitaciones del sistema se hacen evidentes por el hecho de que las marcas con lápiz se han de borrar por completo para que el texto se pueda leer correctamente.

El lector óptico lleva un cable que se introduce en la parte posterior de la tablilla del Omni-reader. A lo largo de la parte trasera también se hallan el conector para fuente de alimentación eléctrica, un conector D RS232C estándar de 25 patillas y dos juegos de interruptores DIP, uno para establecer la velocidad en baudios y otro que ofrece funciones de movimiento manual y espaciado de caracteres.



Formas de ver
Estos cuatro LED indican funciones adicionales. Por ejemplo, POORCOPY permite la lectura de textos de impresión deficiente, mientras que NUMERIC sólo lee números

¿De qué tipo?
Estos cuatro LED indican cada una de las cuatro fuentes de que dispone el dispositivo y se usan conjuntamente con la instrucción TYPEFACE

Representación de instrucciones
Estas instrucciones las puede leer el Omni-reader. Observe los dos cuadrados situados delante de éstas: indican al OCR que lo que sigue es una instrucción en lugar de texto a transferir al ordenador

Regla de posición
La regla permite posicionar la línea de texto con precisión dentro de la ventana de exploración. Observe la franja codificada en la parte inferior de la ventana, que le dice al OCR de qué lado está explorando

Límites exteriores
Estas dos flechas señalan el límite máximo del papel. Más allá el OCR no puede explorar de forma eficaz

Para leer texto en el ordenador, usted primero debe colocar en la tablilla la copia y después alinear una ventana, marcada en la regla, sobre la línea de texto a leer. Luego se pulsa el botón del lector óptico y la unidad se desplaza a lo largo de la regla, explorando el texto a través de la ventana. Al cabo de aproximadamente un segundo, si el Omni-reader ha leído con éxito la línea, el LED se encenderá una vez, sonará un *beep* y la línea aparecerá en la pantalla. Si el lector no ha leído correctamente la línea, el LED parpadeará y se emitirá el *beep* dos veces. El usuario puede entonces optar por volver a someter la línea efectuando otra pasada a través de la pantalla (pulsando el botón de abajo y volviendo a intentarlo), o bien decir al ordenador que acepte la línea incorrectamente leída pulsando el botón de arriba.

El Omni-reader compara el patrón que lee con la lista de caracteres que posee para ese tipo de letra en particular. Los caracteres fuente se retienen en la memoria del Omni-reader en forma de plantillas de bits, y la unidad obtiene los caracteres explorando constantemente la imagen enfocada a través de la lente. La imagen se divide entonces en 50 «reba-

OMNI-READER

INTERFACES

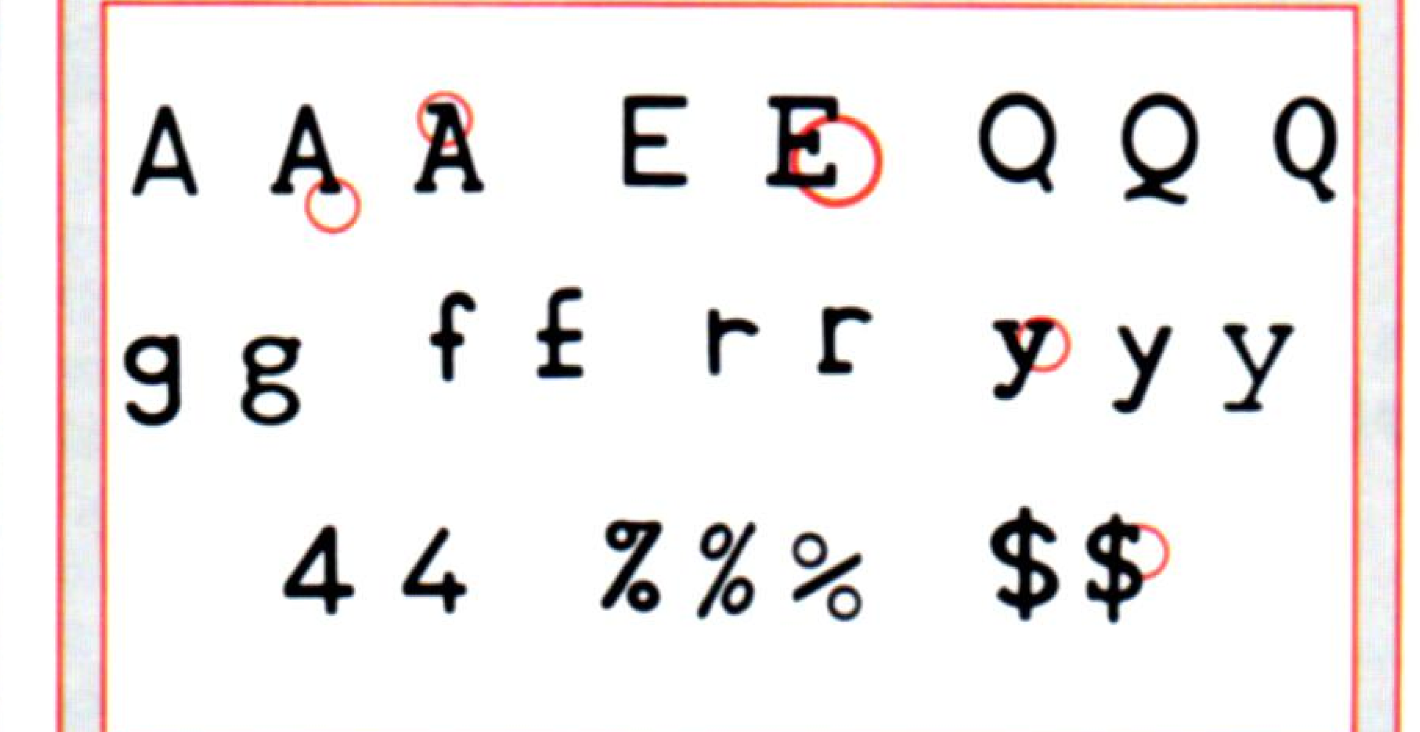
Interface RS232C estándar con velocidad en baudios regulable

VENTAJAS

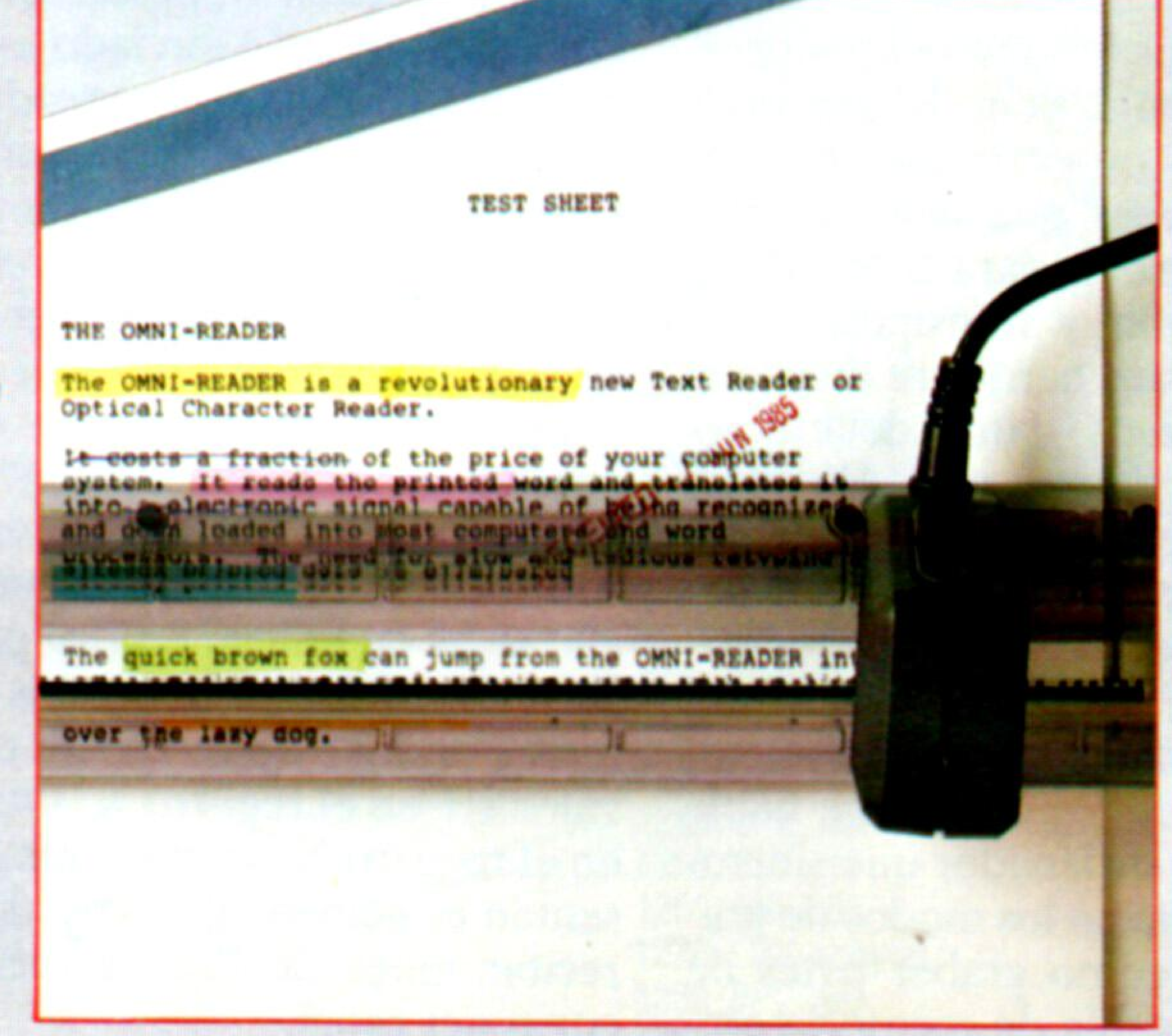
El Omni-reader es veloz y preciso y puede entrar copias en el ordenador de forma mucho más rápida que un mecanógrafo. El dispositivo viene con software y cable de conexión incluidos

DESVENTAJAS

Diversas limitaciones de hardware restringen la lectura precisa a una cantidad limitada de tipos y tamaños de letra. El campo de acción del dispositivo se ve aún más restringido por el hecho de que la tinta haya de tener una base de carbón



Caracteres dudosos
Aunque al observador ocasional le puede parecer que todos los tipos de letra son muy parecidos, existen ciertas diferencias clave que dificultan la detección electrónica de caracteres. Entre éstas, la fundamental son los trazos de pie, extensiones de los caracteres señaladas mediante círculos. Dado que el Omnireader posee plantillas fijas en su memoria, todo lo que no concuerde exactamente con el carácter de la página impresa será rechazado. Sin embargo, aun si pudiéramos proporcionar al carácter alguna clase de «emparejamiento borroso», hay algunas letras, como los ejemplos de la *Q* y la *g*, que son muy diferentes



Una ventana a las palabras
Para que el dispositivo de reconocimiento óptico de caracteres trabaje con eficacia, el texto se debe posicionar exactamente en el centro de la ventana. Abajo de la ventana de exploración hay una serie de bloques, bastante similares a los que se observan en los códigos de barras comerciales. Al explorarlos, éstos indican al Omni-reader la dirección en la que se está desplazando el cabezal del OCR

nadas», tanto en dirección horizontal como vertical.

Al desplazarse a través de la línea de texto, el Omni-reader busca caracteres enteros debajo del cabezal. Sólo procesará el carácter que se halle en el centro de su campo de visión y que no esté cortado por ninguno de los límites de la superficie que se esté explorando. (Éste es el motivo por el cual el carácter se explora tanto horizontal como verticalmente.) El carácter se envía luego para cotejarlo con los caracteres de la memoria del dispositivo.

Cuando el Omni-reader encuentra una pareja, el software envía al ordenador el código ASCII equivalente a través de la interface RS232C y el carácter aparece en la pantalla. Debido a que el Omni-reader sólo puede hallar una pareja que concuerde exactamente con la plantilla retenida en la memoria, el texto ha de estar correctamente alineado en relación a la ventana de exploración. Por consiguiente, si las letras con trazo bajo (como las *y, g, p* y *q*, que poseen una parte del carácter por debajo de la «línea») quedan fuera de la ventana, quizá se las tome por otra cosa; por ejemplo, se puede identificar una *p* por una *o*.

Por el contrario, si la ventana se halla demasiado abajo, puede ser que el explorador óptico tome las partes de arriba de los caracteres de la línea de abajo e intente interpretarlos también. Asimismo surgirán problemas si el usuario intenta entrar texto que sea demasiado largo o bien demasiado corto como para caber con comodidad dentro de la superficie de exploración. Oberon aconseja utilizar solamente caracteres de 10 o 12 puntos (cuatro o cinco caracteres por centímetro). Todo cuando quede fuera de esta gama es susceptible de generar un nivel de error inaceptable.

Manteniéndose dentro de estas restricciones, el Omni-reader de hecho trabaja sorprendentemente bien. El tiempo óptimo de exploración para una línea de texto A4 es de entre 0,5 y 1,5 segundos, que es aproximadamente lo que se tarda en realizar una pasada uniforme a través de la línea. Aunque el usuario novel puede experimentar algunas dificultades iniciales para alinear el texto correctamente, aprender a juzgar el espaciado no lleva mucho tiempo; tras media hora ya se pueden producir líneas sin que se produzcan errores.

Para ayudar al usuario a leer los tipos de letra de menor calidad se han añadido algunas opciones. Éstas se suministran al Omni-reader pasando el lector óptico sobre instrucciones escritas; de éstas, cuatro están situadas en la parte superior de la tablilla y el resto están listadas en el manual. Cada instrucción requiere ir precedida por dos cuadrados sólidos para que sea leída como tal y no como una palabra más a visualizar en la pantalla.

De las cuatro instrucciones incluidas en la tablilla la que se utiliza con mayor asiduidad es TYPEFACE, que permite que usted elija la fuente a leer. NUMERIC suprime de la fuente los caracteres alfabéticos y sólo leerá números, acelerando, por tanto, el proceso de lectura. Quizá esta opción sea una de las aplicaciones más útiles del Omni-reader, puesto que generalmente es al copiar cifras cuando se cometen la mayor parte de los errores no detectados.

La entrada directa es un buen método para obviar este problema. Para tratar con las impresiones de menor calidad se puede emplear la opción POORCOPY, que reduce la velocidad de lectura del Omni-reader de modo que pueda obtener una imagen más clara del carácter situado debajo del lector óptico. Resulta de especial utilidad al leer tipos generados mediante sistemas de cinta de tela entintada, en los cuales el carbón es menos denso.

El software Omni-reader se compone de un par de programas activadores que residen en el área de memoria del sistema. Esto significa que al mismo tiempo se pueden cargar en el ordenador programas de aplicaciones. En consecuencia, los datos leídos de un trozo de papel se pueden entrar directamente en un programa de tratamiento de textos como el *WordStar* o, si usted está leyendo cifras, las puede leer sobre las columnas de una hoja electrónica para obtener cálculos instantáneos.

El precio del Omni-reader incluye el software y un cable de conexión para el ordenador que usted elija. El precio también da derecho a medio día de formación y un año de mantenimiento gratuito a domicilio. No se trata en modo alguno de un dispositivo barato. No obstante, allí donde se reciban muchas copias mecanografiadas, como en la redacción de un periódico, invertir en un Omni-reader a la larga será decididamente más económico que emplear un mecanógrafo a tiempo completo.

Crispin Thomas

Guía visual
El dispositivo se guía de forma manual a través de la línea de texto. Al final de una línea se enciende el LED superior. Si se enciende una vez, la lectura ha sido correcta; si se enciende dos veces es porque el esfuerzo no ha sido totalmente provechoso. Una luz continua indica que el dispositivo no ha conseguido leer la línea entera

Pieza clásica

Concluimos esta serie sobre la interface MIDI analizando otras técnicas para escribir software para la misma

En el capítulo anterior examinamos las diferencias existentes entre la grabación analógica normal de una cinta y la grabación de datos digitales MIDI. Vimos también cómo estas diferencias pueden ser aprovechadas para reducir el empleo de la memoria central por medio de mensajes temporizadores.

La grabación digital y la aplicación del playback o ejecución en diferido nos proporciona un buen ejemplo de algunos problemas que comúnmente encontramos cuando se programa para la MIDI. El programa ha de poder recibir y transmitir datos MIDI por medio de la interface hardware diseñada anteriormente; más aún, esos datos deben ser transmitidos y recibidos en tiempo real. En otras palabras, el programa debe ejecutarse lo suficientemente rápido como para aceptar, procesar y almacenar un mensaje MIDI antes de la llegada del siguiente mensaje.

Además, puesto que estamos grabando en tiempo real, debemos disponer de alguna fuente fiable de temporización dentro del ordenador que sincronice las acciones del programa en los modos de grabación y reproducción, así como grabar bytes de temporización dentro de la tabla de memoria empleada para almacenar la secuencia grabada.

Muchas aplicaciones que exigen actualizaciones periódicas están gestionadas con interrupciones, es decir, emplean señales de interrupción IRQ para ejecutar regularmente secciones determinadas de código de programa. Sin embargo, para esta aplicación, el intervalo entre IRQ sucesivas debe ser más largo que el tiempo entre dos mensajes MIDI sucesivos. Cuando el programa está en la opción grabación, esto puede ser catastrófico, dado que todos los mensajes MIDI enteros pueden perderse, con efectos verdaderamente sorprendentes al reproducir la pieza. Por ejemplo, si un mensaje de *«note off»* (sonido excluido) se perdiera durante la fase de grabación, esa nota o sonido, una vez generada, estaría sonando indefinidamente. Debemos, pues, desactivar las interrupciones IRQ y buscar otra fuente de temporización para nuestro programa.

Temporización para MIDI

Los chips de E/S que emplean el Commodore 64 y el BBC Micro son parecidos y cada uno tiene un registro temporizador de 16 bits incorporado, que puede ser programado directamente para que cuente hacia atrás, desde un determinado valor hasta cero. Si opera así, se dice que el temporizador está funcionando en *modo de libre ejecución*.

El programa utiliza este temporizador como sigue: la subrutina check comprueba si el temporizador se ha «agotado» interrogando al registro flag IRQ del chip CIA/VIA y pone el flag Z en el registro de estado del procesador en el modo conveniente. Las secciones de grabación y reproducción del programa llaman, ambos, a la subrutina check y emplean las instrucciones BNE y BEQ al volver de esta rutina, bifurcando según el estado del flag Z.

La rutina check inspecciona también el teclado para ver si se ha pulsado la barra espaciadora para detener la secuencia. En el Commodore 64 esta inspección se realiza, en ausencia del rastreador de teclado normal gestionado por IRQ, comprobando la última fila de teclas en el teclado. La versión del BBC Micro comprueba el teclado a través de la llamada &79 OSBYTE. Siempre que hagamos que esta rutina check sea llamada al menos una vez entre dos finales de tiempo sucesivos, tendremos un método fiable para temporizar la recepción y transmisión de mensajes MIDI.

No es necesario, ni siquiera recomendable, grabar todo mensaje enviado en MIDI. Dado que el programa *Grabación/reproducción* genera su propia temporización, los mensajes de sistema en tiempo real son irrelevantes y deben ser ignorados si son recibidos. (Muchos instrumentos de teclado son incapaces de transmitir estos códigos sin un secuenciador incorporado.)

Los mensajes de sistema comunes y exclusivos pueden ser emitidos inadvertidamente y deben ser ignorados. Sin embargo, estos mensajes pueden componerse de cualquier número de bytes y por ello se ha de utilizar un flag para instruir a la grabadora de que ignore todos los datos hasta que no se reciba un byte de estado de canal. Esto lo indica el valor $FF en el registro X. Otros valores contenidos en el registro X actúan como contadores que representan el número de bytes de datos que se han de recibir antes de que se complete el mensaje en curso del canal.

Posibles mejoras

Este sencillo programa es esencialmente un ejemplo del tipo de problemas que se encuentran cuando se procesan datos MIDI en tiempo real. Varias son las posibles mejoras que pueden hacerse al programa, que exigen diferentes esfuerzos de programación.

1. Si se detiene la reproducción en una semisecuencia dejará generalmente sonando un número indefinido de notas dado que no se han recibido mensajes de «exclusión de notas».
2. No se ha previsto la sincronización de otros dispositivos tales como máquinas de ritmos y grabadoras de cinta.
3. No es posible editar la secuencia.
4. El tempo de grabación y reproducción es fijo.
5. No se puede silenciar una determinada secuencia.
6. No es posible grabar mientras se visualiza una secuencia diferente.
7. Si el cable MIDI se desconecta durante el mensaje, el programa quedará colgado.
8. No se ha previsto el guardado y la carga de las secuencias grabadas en disco o cassette.



Prog. «Grabación/reproducción»

```
5 REM **   PROYECTO MIDI EN BBC
6 REM **   GRABACION DIGITAL
7 REM **   Y REPRODUCCION
8 *FX 14,6
10 DIM start &2000
20 FOR opt%=0 TO 3 STEP 3
30 timer=&FE60
40 osasci=&FFE3
50 osrdch=&FFE0
60   osbyte=&FFF4
70 streg=&FEE0
80 datreg=streg+1
90 mem=&70
100 ptr=&72
110 P%=start
120 [
130   OPT opt%
140 .begin
150 JMP start1
160   .nobytes BRK
170   .clocks BRK
180   .freemem BRK
190   BRK
200 .start1
210 LDA #3
220 STA streg
230 LDA #&16
240 STA streg
250 .g10
260 LDA #&FF
270 STA 1 owmem
280 LDA #&40
290 STA timer+11
300 LDA #0
310 STA timer+4
320 LDA #&08
330 STA timer+5
340 .g20
350 LDA #0
360 JSR strout
370 .g21
380 JSR osrdch
390 CMP #ASC"E"
400 BNE g22
410 RTS
420 .g22
430   CMP #ASC"R"
440 BEQ g30
450 CMP #ASC"P"
460 BNE g21
470 .g30
480 PHA
490 LDA #1 owmem MOD 256
500 STA mem
510 LDA #1 owmem DIV 256
520 STA mem+1
530 LDA #frbytes MOD 256
540 STA freemem
550 LDA #frbytes DIV 256
560 STA freemem+1
570 SEI
580 LDY #0
590 PLA
600 CMP #ASC"R"
610 PHP
620   BEQ r00
630 .p00
640 LDA #2
650 JSR strout
660 . .p05
670 JSR read
680 CMP #&FF
690 BNE p07
700 PLP
710 JMP c20
720 .p07
730 STA clocks
740 CMP #&F0
750 LDA clocks
760 BEQ p30
770 .p10
780 JSR check
790 BEQ p10
800 DEC clocks
810 BNE p10
820 BCS p05
830 .p30
840 JSR read
850 PHA
860 .p35
870 LDA streg
880 AND #2
890 BEQ p35
900 PLA
910 STA datreg
920 BPL p50
930 JSR getno
940 .p50
950 DEX
960 BNE p30
970 LDX nobytes
980 BPL p05
990 .r00
1000 LDA #1
1010 JSR strout
1020 STY clocks
1030 .r05
1040 LDX #&FF
1050 .r10
1060 JSR check
1070 BEQ r20
1080 INC clocks
1090 LDA clocks
1100 CMP #&F0
1110 BCC r20
1120 JSR store
1130 STY clocks
1140 .r20
1150 LDA streg
1160 AND #1
1170 BNE r40
1180 CPX #1
1190 BMI r10
1200 BPL r20
1210 .r40
1220 LDA datreg
1230 BMI r50
1240 CPX #0
1250 BMI r10
1260 BNE r80
1270 LDX nobytes
1280 BPL r60
1290 .r50
1300   CMP #&F8
1310   BCS r20
1320   CMP #&F0
1330 BCS r05
1340 JSR getno
1350 .r60
1360 PHA
1370 LDA clocks
1380 JSR store
1390 STY clocks
1400 PLA
1410 .r80
1420 JSR store
1430 DEX
1440 BEQ r10
1450 BNE r20
1460 .check
1470 STX &74
1480 PHP
1490 LDA #&79
1500 LDX #&E2
1510 JSR osbyte
1520 LDY #0
1530 PLP
1540 TXA
1550 BPL c40
1560 PLA
1570 PLA
1580 PLP
1590 BNE c20
1600 LDA #&FF
1610 STA (mem),Y
1620   .c20
1630 CLI
1640 JMP g20
1650 .c40
1660 LDX &74
1670 LDA timer+13
1680 AND #&40
1690 STA timer+13
1700 RTS
1710 .store
1720 STA (mem),Y
1730 INC freemem
1740 BNE point
1750 INC freemem+1
1760 BNE point
1770 PLA
1780 PLA
1790 PLA
1800 LDA #3
1810 JSR strout
1820 JMP c20
1830 .read
1840 LDA (mem),Y
1850 .point
1860 INC mem
1870 BNE p20
1880 INC mem+1
1890   .p20 RTS
1900 .getno
1910 LDX #2
1920 CMP #&C0
1930 BCC n10
1940 CMP #&E0
1950 BCS n10
1960 DEX
1970 .n10
1980 STX nobytes
1990 INX
2000 RTS
2010 .strout
2020 ASL A
2030 TAX
2040 LDA messtab,X
2050 STA ptr
2060 LDA messtab+1,X
2070 STA ptr+1
2080 LDY #0
2090 .m10
2100 LDA (ptr),Y
2110 JSR osasci
2120 INY
2130 CMP #&0D
2140 BNE m10
2150 LDY #0
2160 .m70 RTS
2170 ]
2180 mess0=P%
2190 $P%="R = grabar,P = reproducir, E = salir"
2200 P%=P%+LEN($P%)+1
2210 mess1=P%
2220 $P%="grabacion"
2230 P%=P%+LEN($P%)+1
2240 mess2=P%
2250 $P%="reproducción"
2260 P%=P%+LEN($P%)+1
2270 mess3=P%
2280 $P%="no queda memoria"
2290 P%=P%+LEN($P%)+1
2300 1 owmem=P%+8
2310 frbytes=&E000+(1 owmem-start)
2320 [
2330 .messtab
2340 ]
2350 NEXT
2360 ?P%=mess0 MOD 256
2370 P%?1=mess0 DIV 256
2380 P%?2=mess1 MOD 256
2390 P%?3=mess1 DIV 256
2400 P%?4=mess2 MOD 256
2410 P%?5=mess2 DIV 256
2420 P%?6=mess3 MOD 256
2430 P%?7=mess3 DIV 256
2440 CALL begin
```

Tortuga pequeña

Compararemos la versión Amstrad de Dr LOGO con la producida para el IBM PC

Cuando se transfirió por primera vez el LOGO desde ordenadores centrales a microordenadores de ocho bits, hubieron de introducirse varias restricciones y simplificaciones con el fin de que se pudiera ejecutar el lenguaje con la memoria disponible, de 64 K o incluso menos. Cuando comenzaron a aparecer micros de 16 bits, con memorias mucho mayores, enseguida salieron al mercado numerosas versiones mejoradas de LOGO.

Gary Kildall (el fundador de Digital Research, productora del sistema operativo CP/M) se sintió favorablemente impresionado por el potencial del LOGO como lenguaje y produjo su propia versión para el IBM PC, a la que llamó Dr LOGO. Desde entonces esta versión se ha puesto al alcance de otras varias máquinas de 16 bits. El Apricot F1e, por ejemplo, se suministra ya con Dr LOGO.

Desde entonces Digital Research ha adaptado Dr LOGO para numerosas máquinas CP/M de ocho bits. Amstrad está proporcionando esta versión de Dr LOGO junto con su paquete de disco para el CPC 464.

Dr LOGO, en su implementación para el IBM PC, requiere para su ejecución al menos 192 K de memoria. El disco no se puede copiar porque está protegido, pero se proporciona una copia de seguridad. Para ejecutarlo, se debe insertar el disco y efectuar una inicialización del sistema. La resolución de gráficos en color es de 320 por 200 pixels, con una selección de 16 colores de fondo, junto con una opción de cuatro juegos de tres colores de lápiz (primer plano).

Con una placa de gráficos a color en el PC, junto con una pantalla tanto en color como monocromática, Dr LOGO puede producir gráficos en la pantalla en color y texto en la pantalla monocromática. No obstante, si usted posee una sola pantalla, se pueden mezclar ambos en ella

En el manual se describe a Dr LOGO como una ampliación del LOGO Apple y en realidad contiene todas las instrucciones de LOGO que contiene aquél. Éstas incluyen las instrucciones estándares para gráficos, proceso de listas y gestión del espacio de trabajo, así como primitivas para tratamiento de errores, «propiedades» (una forma de asignarle a un objeto más de un valor) y «paquetes» (una forma de empaquetar procedimientos entre sí en forma de grupo). Dr LOGO es «sensible a los tipos de letra» y, por tanto, las instrucciones del LOGO se deben entrar en minúsculas.

El editor hace uso de las teclas de función IBM, pero también reconoce los mismos códigos de control utilizados en el LOGO Apple. Si se encuentra un error durante la ejecución de un programa, digitan-

Comparando

PROCEDIMIENTO		RESULTADOS			
		IBM PC (256 K)	AMSTRAD CPC 464/664	COMMODORE 64	APRICOT F1e
RECUR	Nivel alcanzado	512	187	90	832
DEFVARS		1307	263	553	1606
ENDRECUR1		512	189	INFINITO	INFINITO
ENDRECUR 2		151	129	65	72
LINES (con tortuga escondida)	Tiempo (al segundo más próximo)	9	53	13	10
LINES (con tortuga escondida)		5	39	11	6
ASSIGN		25	22	6	20
LISTS		64	74	18	69
ARITH		80	138	36	77
RECURA		92	91	24	72

Procedimientos comparativos

Los nueve procedimientos de comparación están diseñados para probar diferentes características del LOGO y para mostrar los puntos fuertes y los puntos débiles de cada implementación.

- RECUR mide el tamaño de la pila
- DEFVARS mide el espacio disponible para definir variables y procedimientos
- ENDRECUR1/ENDRECUR2 determinan si se ha implementado eficazmente la recursión final. Una implementación ideal permitiría la recursión infinita. Mientras que ENDRECUR2 produce su resultado y, en consecuencia, se puede considerar como una operación, ENDRECUR1 es una instrucción
- LINES mide la velocidad de dibujo. Se dan dos cifras para indicar la mejora que se obtiene en cuanto a velocidad al esconder la tortuga
- ASSIGN mide la velocidad a la cual se pueden asignar valores a las variables
- LISTS mide la velocidad de los operadores para proceso de listas
- ARITH mide la velocidad de las funciones aritméticas
- RECURA mide la vel. de las llamadas recursivas

do *ed* se llamará al editor y éste aparecerá ya con el procedimiento que generó el error, listo para editarlo. Lamentablemente, no hay ninguna forma de conocer el punto exacto del error mirando a la pantalla cuando uno se halla en la modalidad de edición. La mayoría de las otras versiones del LOGO indican esto mediante la inclusión de una línea especial en la parte inferior de la pantalla.

Además de las primitivas para proceso de listas habituales, Dr LOGO posee algunas otras nuevas:

sort clasifica una lista por orden alfabético
shuffle ordena de forma aleatoria los elementos de una lista
piece permite seleccionar una parte de una lista

Dr LOGO añade otras para depuración:

watch permite ver el efecto del procedimiento ejecutándolo línea a línea
trace imprime los nombres de las variables a medida que se las va llamando y definiendo
debug divide la pantalla en dos ventanas. La ventana *debug* se puede utilizar para visualizar información proporcionada por "trace" y "watch", mientras que la ventana *program* muestra la salida del programa

A los procedimientos se les puede añadir comentarios, que después se pueden suprimir de todos los procedimientos en el espacio de trabajo, mediante noformat, si así fuera necesario para proporcionar espacio extra para la ejecución.

Existen, asimismo, algunas primitivas nuevas para administración de procedimientos:

follows permite definir el orden por el cual se deben presentar los procedimientos en la pantalla
potl visualiza nombres de procedimientos no llamados desde ningún otro procedimiento
poref< nombre> visualiza nombres de procedimientos que llaman al procedimiento <nombre>
pocall < nombre> visualiza nombres de procedimientos llamados por el procedimiento <nombre>

Con estas facilidades adicionales se pretende ofrecer un entorno mejorado para el desarrollo de programas, en el cual crear programas en LOGO.

Dr LOGO posee algunos puntos débiles: el editor no posee ninguna función de búsqueda ni de sustitución, no hay ningún método de acceder al código máquina y ninguna gestión de archivos. Sin embargo, el manual de 300 páginas es muy claro y exhaustivo. Posee una introducción al LOGO, así como un manual de referencia que incluye una página separada dedicada a cada primitiva.

Evidentemente, Dr LOGO posee numerosas características extras inexistentes en las versiones más pequeñas del lenguaje, todo lo cual crea un entorno para desarrollo de programas muy mejorado.

Es interesante comparar Dr LOGO, desde el punto de vista de espacio y velocidad, con las versiones de ocho bits del lenguaje. El espacio de trabajo del LOGO se mide por *nudos*. Si se le pregunta a Dr LOGO cuántos nudos libres tiene al arrancar el sistema, la alentadora respuesta será de alrededor de 10 000 (típicamente, los sistemas de ocho bits poseen entre 2 000 y 3 000). De modo, pues, que hay muchísimo espacio de trabajo para definir procedimientos y para ejecutar procedimientos recursivos. No obstante, la recursión final (caso de un procedimiento recursivo en el cual la llamada recursiva se halla en la última línea) no está implementada eficazmente, y, en consecuencia, programas que en las máquinas Commodore o BBC se ejecutarían hasta el infinito, en Dr LOGO se quedan sin espacio luego de pocos cientos de llamadas recursivas.

En cuanto a velocidad, los gráficos son rápidos, pero la aritmética y el proceso de listas se llevan a cabo más lentamente que con otras versiones de ocho bits del lenguaje.

La versión Amstrad

Dr LOGO se suministra con la unidad de disco Amstrad y se halla en la cara opuesta del disco CP/M. El Dr LOGO del Amstrad está modificado para sacar partido del hardware en el que se ejecuta. Los gráficos utilizan el mismo sistema de cuatro lápices que el Dr LOGO para el IBM PC, pero en este caso los colores del lápiz se establecen dando una lista de números que determinan las proporciones de rojo, verde y azul requeridas. La versión IBM no posee una amplia gama de instrucciones de sonido, mientras que la versión para el Amstrad posee env (envoltura de volumen), ent (envoltura de tono) y release, que libera canales establecidos en estado de sostenimiento en una instrucción de sonido.

La versión Amstrad posee la mayoría de las facilidades del LOGO Apple, incluyendo propiedades y tratamiento de errores, pero no paquetes.

Las primitivas de depuración y las facilidades extras para administración de procedimientos del Dr LOGO del IBM PC no se han incluido, presumiblemente, debido a la menor capacidad de memoria del Amstrad. Más grave, no obstante, es el hecho de que no haya ninguna función define y aparentemente ninguna forma de suprimir un archivo sin salir al CP/M y luego volver otra vez.

La capacidad de respuesta del editor es muy lenta y puede ser que el usuario se encuentre digitando por adelantado y perdiendo el carácter suelto. Por otra parte, si se encuentra un error al ejecutar un procedimiento, el editor, una vez entrado, posiciona el cursor sobre la palabra que generó el error en el procedimiento, lo que realmente es una característica muy útil.

La versión Amstrad proporciona un acceso limitado al código máquina a través de .examine y .deposit (PEEK y POKE), de las que carece la versión para el IBM PC.

Se entrega una documentación mínima, remitiendo al usuario a un manual de formación y de referencia que aún no se ha publicado. Mientras tanto, éste tendrá que contentarse con el resumen de 25 páginas de las instrucciones. Asimismo, parece haber algunas facilidades de Dr LOGO que se han implementado pero que no se mencionan.

En la versión para el Amstrad han desaparecido muchísimas de las características más interesantes del Dr LOGO para el IBM PC, a pesar de lo cual continúa siendo una buena implementación estándar de LOGO, con extras tales como empaquetamiento y tratamiento de errores.

Tras el encendido, se afirma que Dr LOGO posee 2 105 nudos, cifra típica para los sistemas de ocho bits. Sin embargo, la recursión final no se ha implementado eficazmente; y en cuanto a gráficos, aritmética y proceso de listas, éste es uno de los LOGO de ocho bits más lentos, consumiendo la mayoría de las operaciones alrededor de tres veces más tiempo que en el Commodore 64.

Introducción

Iniciamos esta serie dedicada al FORTH examinando dos características clave de este lenguaje: la interactividad y la ampliabilidad

Inventado a finales de los años sesenta por el astrónomo Charles Moore, el FORTH se diseñó específicamente para controlar un dispositivo científico que necesitaba un posicionamiento de precisión. La utilidad del lenguaje va, sin embargo, mucho más allá de las aplicaciones de control, pero a modo de introducción a sus principios operativos básicos es interesante observar cómo puede ser utilizado para controlar un telescopio. Sería gratificante sentarse en el observatorio frente a un terminal de ordenador y, por ejemplo, digitar:

```
30 GRADOS ELEVACION
```

y el telescopio se moviera inmediatamente y con toda precisión a dicho punto. También sería práctico poder empaquetar y almacenar secuencias útiles de instrucciones para economizar la digitación repetitiva. Por ejemplo, si la posición de estacionamiento fuera de 90° de elevación y 0° de acimut (términos que describen el ángulo y la posición del telescopio), usted podría describir una nueva instrucción, PARK, como:

```
0 GRADOS ACIMUT 90 GRADOS ELEVACION
```

En resumen, usted necesitaría:

- Instrucciones elementales para el telescopio.
- Formas de combinar las instrucciones elementales en otras más complejas: «programas» para telescopio.

Este sistema equivaldría a un lenguaje de programación completo para telescopios. Sin embargo, requeriría una propiedad extra, bastante especial: habría de aceptar instrucciones (tanto las incorporadas como ELEVACION y otras nuevas) directamente desde el teclado. Además, habría de hacer todo esto de la manera más simple posible.

Intente imaginar lo que sería satisfacer estas exigencias en BASIC, que es bastante bueno en cuanto a permitir que usted utilice sus instrucciones incorporadas, como PRINT y RUN, desde el teclado. Las instrucciones nuevas, sin embargo, serán un problema. Suponiendo que ya hubiera definido ACIMUT y ELEVACION como rutinas GOSUB en las líneas 1000 y 1100, podría escribir una instrucción para PARK como:

```
1200 REM PARK
1205 LET GRADOS=0
1210 GOSUB 1000
1215 LET GRADOS=90
1220 GOSUB 1100
1225 RETURN
```

O quizá pudiera predefinir una variable PARK mediante:

```
LET PARK = 1200
```

y posteriormente impartir la instrucción (en algunas versiones de BASIC):

```
GOSUB PARK
```

Probablemente lo mejor que podría hacer usted fuera definir la rutina GOSUB como un procedimiento (como en el BASIC BBC) y digitar algo como:

```
PROCPARK
```

Por tanto, siempre que llame a una de sus propias rutinas de telescopio, debe entrar instrucciones adicionales como GOSUB o PROC así como el nombre o el número de línea de su rutina.

Una alternativa es escribir un intérprete, un programa que tome instrucciones para el telescopio y las ejecute, algo así como:

```
100 INPUT C$
150 IF C$ = "PARK" GOSUB 1200:GOTO 100
```

Pero entonces usted ya no podrá utilizar las instrucciones del BASIC (como PRINT) de forma directa.

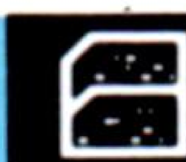
Estamos ahora en condiciones de apreciar que las dos propiedades de nuestro lenguaje para telescopio serán:

- En primer lugar, ha de ser *interactivo*. Debe ser capaz de ejecutar sus instrucciones apenas usted las digite. Esto es como en BASIC, LOGO, APL y PROLOG, pero no como en PASCAL, ALGOL, FORTRAN y COBOL. También algo parecido al módulo de interpretación de comandos de algunos sistemas operativos, como el CP/M y el Unix.
- En segundo lugar, ha de ser *ampliable*. El forma-



Para iniciarse en el lenguaje
El Jupiter Ace fue un valiente intento por comercializar una máquina que como lenguaje estándar ofreciera FORTH en lugar de BASIC. Lamentablemente, la falta de facilidades de color en el Ace, junto con una pequeña memoria, el desconocimiento del FORTH y la fuerte competencia del Sinclair Spectrum lo colocaron en una situación de evidente desventaja respecto a otros micros de su categoría. Sin embargo, una máquina de segunda mano junto con la documentación original (que era de un nivel muy elevado) proporcionarían una excelente introducción al lenguaje

Ian McKinnell



to que haya definido usted para llamar a las instrucciones debe ser idéntico al empleado para las instrucciones incorporadas. Luego, tras haber definido sus propias rutinas, será como si usted estuviera utilizando un lenguaje ampliado, más poderoso, y no el lenguaje original más algunas rutinas nuevas. Esto es como en LOGO y los sistemas operativos, pero no como en ninguno del resto de los lenguajes.

El concepto de ampliabilidad en realidad es aún más importante de lo que pueda desprenderse de lo dicho hasta ahora. Si, por ejemplo, alguien nos vende un lenguaje para control de telescopios y nosotros le añadimos pequeñas ampliaciones para que incluya rutinas útiles para nuestro propio telescopio, habremos ampliado un «lenguaje para control de telescopios de propósito general» original, convirtiéndolo en un nuevo «lenguaje personalizado para control de telescopios». Pero para comprender cabalmente el concepto de ampliabilidad, deberíamos poder empezar con un «lenguaje de ordenador ampliable de propósito general» y ampliarlo hasta convertirlo en un «lenguaje para control de telescopios de propósito general» (en realidad, el mismo se podría ampliar a lenguajes para controlar robots, experimentos científicos, cadenas de producción o cualquier dispositivo que se pueda conectar a un ordenador). Además, no es necesario que lo que se controle sea físico. Podría ser un concepto abstracto interno del ordenador, como la tortuga en una pantalla de gráficos tortuga, los registros en un sistema de archivo o, de hecho, cualquier cosa que usted deseara manipular por medio de un programa.

De los enfoques de que disponemos, el FORTH es el que más se aproxima a un «lenguaje de ordenador ampliable de propósito general». Hemos hablado de los problemas especiales (interactividad y ampliabilidad) que el FORTH intenta resolver, y en nuestro próximo capítulo comenzaremos a hablar de sus soluciones.

Dialectos de FORTH

Existen tres dialectos principales de FORTH: fig (FORTH Interest Group) FORTH, FORTH-79 y FORTH-83. La mayoría de las implementaciones se describen de acuerdo a alguna de estas tres. El FORTH-79 y el FORTH-83 son estándares sucesivos y especifican los juegos mínimos de instrucciones y rutinas que han de estar disponibles. Casi con toda seguridad las implementaciones individuales le ofrecerán más, pero si usted escribe un programa ateniéndose estrictamente al FORTH-83, por ejemplo, lo podrá ejecutar en todas las implementaciones FORTH-83, aun cuando no haga un uso completo de todas las facilidades disponibles. El FORTH-83 es el estándar actual y nos atendremos a él. El FORTH-79 en líneas generales es igual, con muy pocas incompatibilidades. También posee numerosas facilidades extras, incorporadas al construirse la implementación, que sus usuarios tienden a considerar como estándares. Muchas implementaciones del FORTH-79 y el FORTH-83 son versiones del figFORTH modificadas para adecuarlas al estándar

El telescopio y la tortuga

El FORTH y el LOGO son lenguajes muy similares. Ambos se diseñaron originalmente para controlar objetos físicos: un telescopio y una tortuga. Además tienen en común dos conceptos operativos fundamentales: una rutina se llama simplemente entrando su nombre, y se definen rutinas nuevas utilizando rutinas ya existentes. La principal diferencia entre los dos lenguajes proviene de las distintas expectativas de los usuarios a los que están dirigidos. Los usuarios de FORTH exigen velocidad de ejecución (de modo que el lenguaje no sea sólo interactivo y ampliable sino también eficaz) y para conseguirlo están dispuestos a admitir algunas incomodidades. El LOGO, por otra parte, está dirigido básicamente a ser utilizado por niños en edad escolar y no permite tales concesiones. En consecuencia, el LOGO opera con mucha más lentitud. Comparemos los dos lenguajes en mayor profundidad. El FORTH no tiene incorporados gráficos de tortuga como estándar, pero posee facilidades para manipular la pantalla terminal. Usted puede utilizar estas dos facilidades de bajo nivel para definir las palabras de tortuga FORWARD, RIGHT, etc., creando, por tanto, una versión «amplia» del FORTH que incluya gráficos de tortuga. Consideremos estas formas alternativas de controlar a la tortuga en los dos lenguajes:

LOGO	**FORTH**
FORWARD 100	100 FORWARD

A partir de este ejemplo, podría parecer que el FORTH hace las cosas al revés. Pero cuando una rutina en FORTH necesita parámetros, usted primero debe calcular estos parámetros y escribirlos antes del nombre de la rutina. Esto se podría considerar como una técnica de «libro de recetas»: primero reunir los ingredientes (en cantidades especificadas) y luego cocinarlos. Otras correspondencias con el LOGO incluyen la definición de rutinas nuevas:

TO...END	:...;

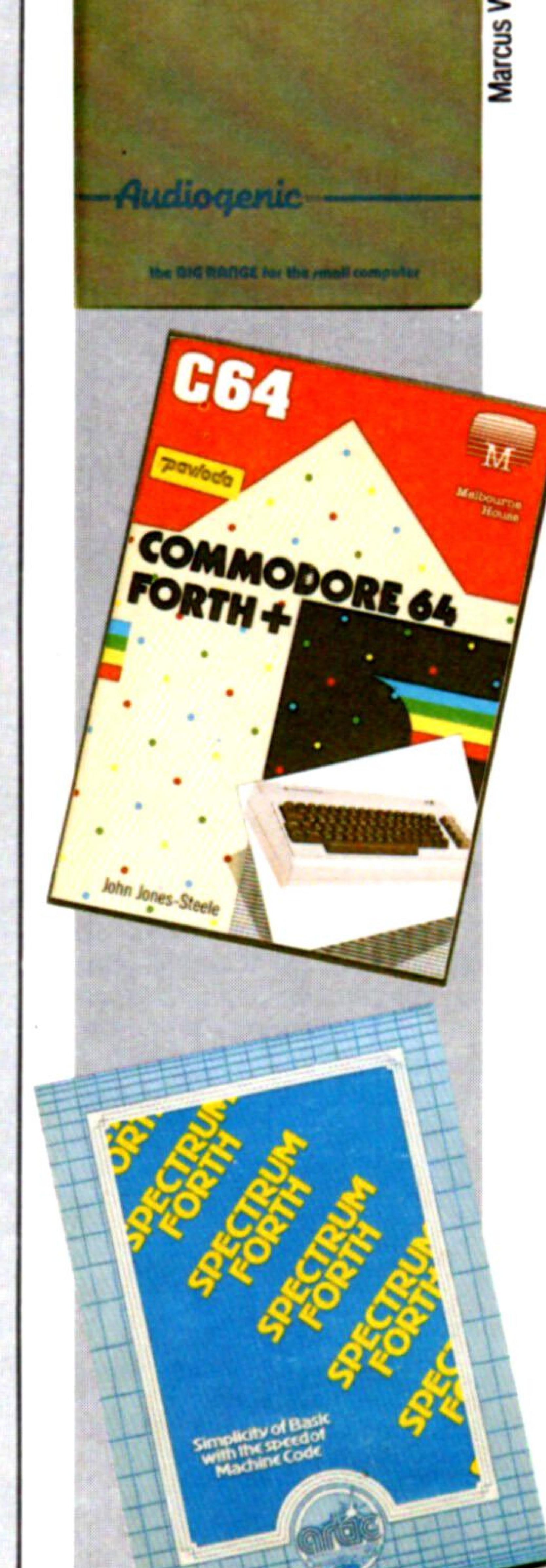
y estructuras de bucle:

REPEAT 4[...]	4 DO...LOOP

He aquí cómo podríamos combinar las dos en la definición de un procedimiento CUADRADO:

```
TO CUADRADO LADO        :CUADRADO
REPEAT 4[                4 DO
   FORWARD LADO            DUP ADELANTE
   RIGHT 90                90 DERECHA
   ]                       LOOP
                           DROP
END                      ;
CUADRADO 100               100 CUADRADO
```

DUP y DROP son «manipulaciones de pila» que permiten que las rutinas accedan al parámetro de CUADRADO (DUP la reproduce, haciendo una copia para ADELANTE;DROP la «abandona»; en otras palabras, la elimina de modo que no se cruce en el camino del siguiente cálculo)



Marcus Wilson-Smith

De especial interés...
Existen implementaciones de FORTH para muchos micros personales. Entre los paquetes más accesibles se incluyen: FORTH Acornsoft (también disponible en ROM) y SkyWays MultiFORTH (con multitareas) para el BBC Micro, y Abersoft FORTH y Artic FORTH para el Sinclair Spectrum. El FORTH es el único lenguaje de ordenador que ha sido difundido por sus usuarios (que originalmente se enviaban implementaciones a través de Clubes de Usuarios). En Londres existe un floreciente Club de Usuarios de FORTH. El grupo se reúne a las 7 de la tarde el primer jueves de cada mes en el South Bank Polytechnic, Borough Road, London SE1. Usted puede asociarse (la suscripción anual es de £ 7) escribiendo a:
FORTH Interest Group
7 Wyndham Cresent
Woodleigh
Reading
RG5 3AY7719
Gran Bretaña

Todo un serial

El análisis general que hemos llevado a cabo de las rutinas de la Interface 1 exige que nos ocupemos ahora de los códigos de enganche con los que controla la puerta serial RS232

Comenzaremos por dar una visión general de una interface serial, con un enchufe RS232 de 270° y nueve patillas (el dibujo ilustra la función de cada patilla). El número de estas líneas necesarias para la conexión depende de la complejidad de la aplicación, pero un enlace mínimo puede establecerse con las patillas 2, 3 y 7. Si usted hace esto las patillas 4 y 5 han de estar conectadas a la patilla 9 para mantenerlas en 1 lógico. Esto evita que el Spectrum quede colgado si, estando lo que sea en el otro extremo, el enlace todavía no se halla preparado para recibir datos desde el ordenador. Esto tiene la desventaja, sin embargo, de que se pierden los datos si el otro extremo del enlace no está disponible para recibirlos. En general, es preferible el empleo del cableado completo.

Una aplicación posible del enlace RS232 es la conexión del Spectrum a una impresora *real*. Una vez desplegado el cable, el software de control muestra un manejo muy sencillo. Para enviar texto a la impresora, establecemos la velocidad en baudios desde el BASIC por medio de FORMAT, y ejecutamos después el siguiente código:

```
CLOSE #3
OPEN #3;"T"
```

Esto acopla la corriente 3, que es la corriente usual de impresora en el Spectrum, al canal T, que es el canal de texto del enlace RS232. El manual de la Interface 1 da más detalles sobre el empleo de las corrientes *texto* y *binaria*. LLIST y LPRINT enviarán ahora los datos al dispositivo ajustado a la Interface 1.

Veamos ahora algunos detalles de cómo opera la interface serial. Los datos se transmiten con un bit de inicio, ocho bits de datos y dos bits de parada. No hay por qué preocuparse por este formato, dado que las rutinas del sistema operativo del Spectrum ya se encargan de ello. Lo que más importa es la velocidad en baudios a la que son enviados los datos. Esta velocidad suele establecerse con la instrucción FORMAT, que nos limita a una gama determinada de velocidades. Aunque hay poca pérdida de la integridad de los datos con una interface RS232, incluso a considerable distancia, siempre es bueno emplear velocidades más bajas de transmisión según aumenta la distancia entre el emisor y el receptor.

Es un procedimiento recomendable insertar variables de sistema de la Interface empleando el código de enganche 49 antes de usar cualquiera de las siguientes técnicas. Hay dos códigos de enganche referidos al RS232 y tres variables de sistema. Veamos primero las variables.

- BAUD (posiciones 23747 y 23748) es una variable de dos bytes que contiene un valor que determina la velocidad en baudios que va a emplearse en la transmisión y recepción de datos. Esto representa una de las limitaciones de la interface del Spectrum: su incapacidad de manejar dos velocidades de transmisión distintas para enviar y para recibir.
- SERFL (posición 23751) es un byte flag, empleado por el OS sólo al recibir. Si está a uno, quiere decir que hay todavía un byte de lectura disponible.
- SERBYT (posición 23752) actúa como un buffer de entrada de un byte en recepción. Ocasionalmente es posible recibir y almacenar un byte en esta posición después de que usted haya cortado la comunicación. Pueden presentarse problemas con el empleo de las dos últimas variables, como veremos.

Una cuarta variable de sistema, la IOBORD (posición 23750) no se refiere estrictamente a la interface serial, sólo contiene el color del marco de la pantalla (*border*) que se emplea en las operaciones de E/S. Por tanto, puede hacer que el marco contenga cualquier color durante estas operaciones.

Cuando queremos emplear la interface RS232, lo primero que hay que hacer es establecer la velocidad de transmisión. Esto se hace dando un valor apropiado a BAUD, que, en realidad, es todo lo que hace FORMAT. El valor a usar para una determinada velocidad se calcula así:

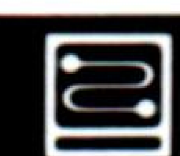
Velocidades baudio con la Interface 1

La RS232 estándar se emplea con profusión en las comunicaciones, en especial cuando la transmisión de datos se realiza a larga distancia, y la *velocidad baudio* proporciona una indicación del número de bits de datos por segundo que se transmiten. Con la fórmula de que se habla en el texto aplicada a algunas de las más comunes densidades obtenemos los valores siguientes:

Velocidad baudio	Valor
50	2690
110	1221
300	446
600	222
1200	110
2400	54
4800	28
9600	12
19200	5

El valor de la velocidad en baudios deseada se coloca (POKE) en la variable de sistema BAUD en las direcciones 23747 y 23748. El que exista una sola variable significa, por desgracia, que debe usarse la misma velocidad de transmisión para la emisión y para la recepción. El siguiente fragmento en código máquina ilustra el empleo de estos valores. Para establecer una velocidad de 300 baudios:

```
LD     HL,446
LD     (23747),HL
RET
```



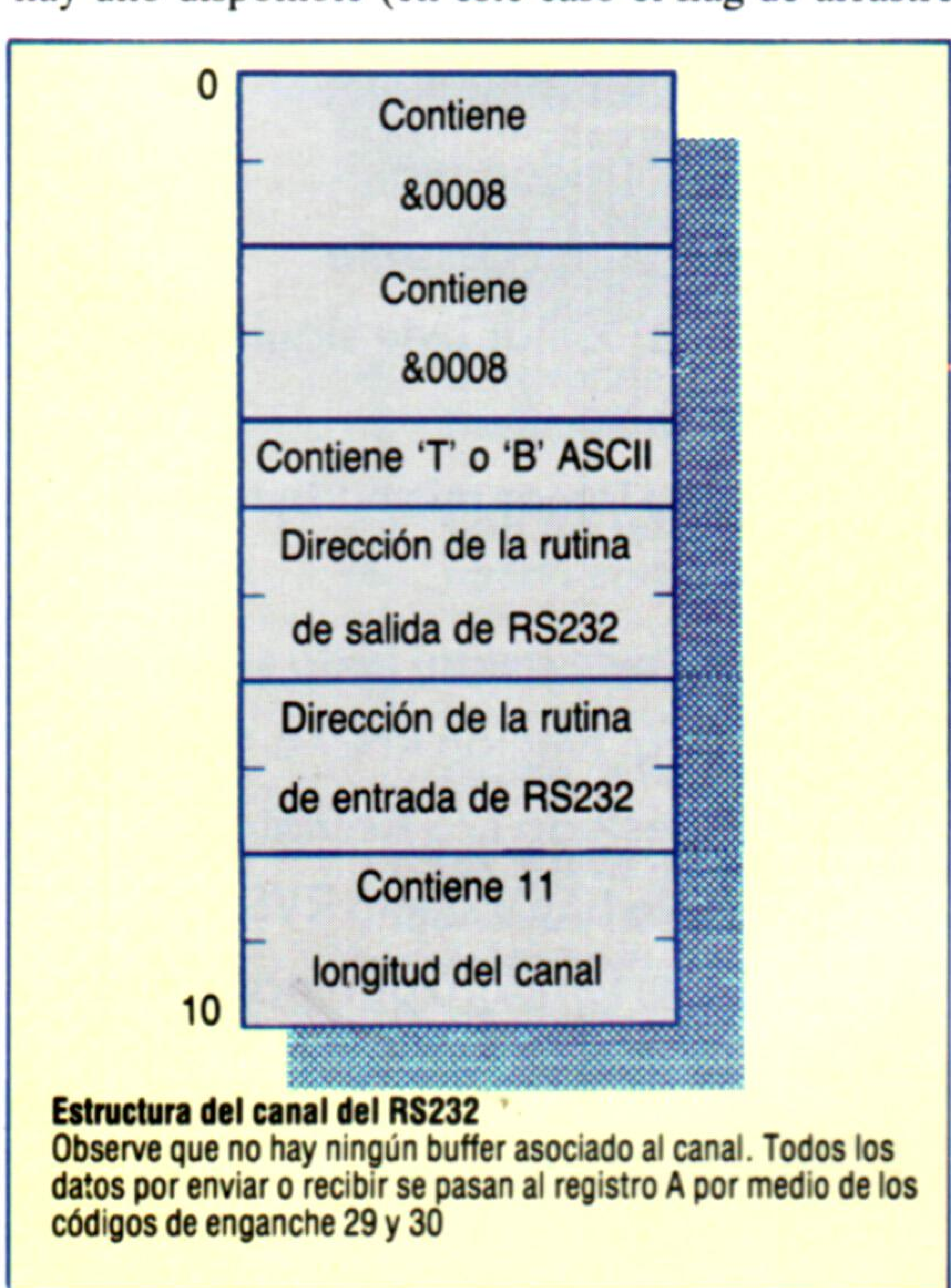
valor = (3500000 / (26 * velocidad en baudios))−2

Con esta ecuación es posible establecer la interface serial en una velocidad en baudios no habitual desde el código máquina, pero quizá no tenga utilidad alguna si usted no está comunicando con otro Spectrum a través de la RS232. La tabla de velocidades proporciona varios valores y la rutina necesaria para su implementación.

Veamos ahora el modo como son transferidos los datos a través del enlace. En BASIC, la apertura (OPEN) de una corriente a emplear con la interface RS232 genera un canal de la forma mostrada en el gráfico *Estructura del canal del RS232*. Este canal está situado en el área de canales de la memoria. El byte 4 de este canal tiene añadido 128 a su valor si el canal ha sido incrementado implícitamente (es decir, por medio de SAVE*, MOVE, FORMAT, etc.).

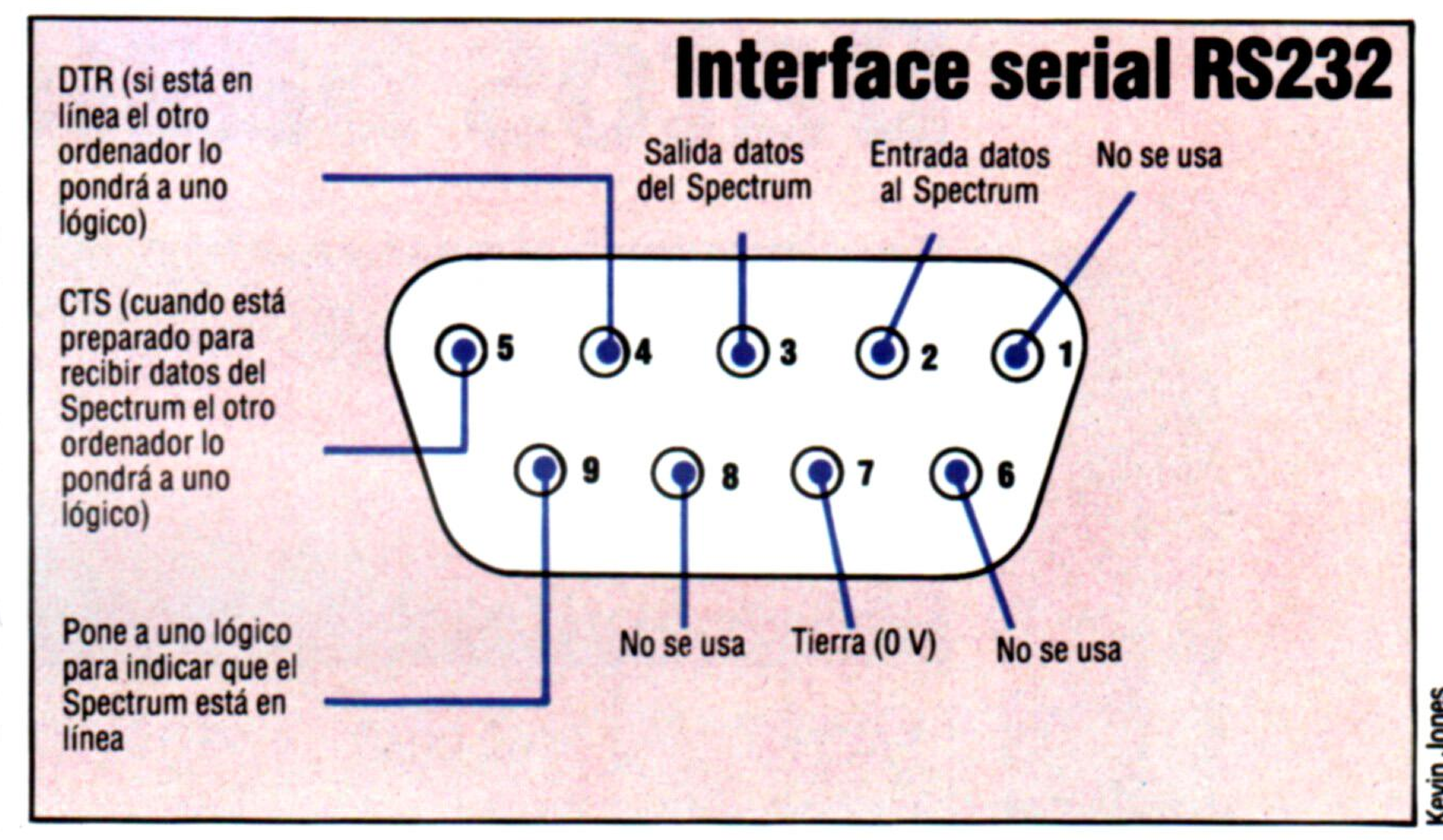
Pero esto sólo tiene un interés académico para nosotros cuando lo que vamos a emplear son las rutinas del RS232 desde el código máquina. Note que no existe ningún buffer asociado a tal canal. Escribimos y leemos datos llamando al correspondiente código de enganche, que a su vez llama a la rutina cuya dirección está o bien en las posiciones de canal 5 y 6 (transmisión) o bien en la 7 y 8 (recepción). No hay establecimiento de canal cuando se emplea un código de enganche. Los dos códigos de enganche del RS232 son:

- *Código de enganche 29:* Cuando se le llama da un único byte de la interface serial en el registro A si hay uno disponible (en este caso el flag de arrastre se pondrá a uno). Si no hay un byte disponible, A contendrá un 0 y el flag se pondrá a 0.



Estructura del canal del RS232
Observe que no hay ningún buffer asociado al canal. Todos los datos por enviar o recibir se pasan al registro A por medio de los códigos de enganche 29 y 30

- *Código de enganche 30:* Transfiere el byte contenido en el registro A a la interface serial para su transmisión según la velocidad en baudios establecida. Al emplear este código, todos los bytes son transmitidos fielmente: no hay diferencia entre las corrientes B y T a nivel de lenguaje máquina.



Y es en este punto donde los problemas relacionados con SERBYT y SERFL pueden surgir. Si la rutina del código de enganche encuentra SERFL puesta a uno, tomará un byte de SERBYT, aun cuando ese byte haya sido «escondido» en SERBYT desde que se completó el último impulso de la comunicación serial. Este «byte polizón» exige ser atendido si aparece al comienzo de una serie de bytes recibidos; todos los bytes siguientes pueden ser recibidos, pero con un significado alterado. La manera más fácil de librarse de lo que pueda haber en SERBYT consiste simplemente en poner SERFL a cero antes de tratar de leer cualquier cosa desde la interface serial. Un sencillo código máquina lo hará:

```
XOR     A
LD      (23751),A
```

Ésta es la única complicación efectiva que mucha gente puede encontrar cuando emplea la interface serial, aunque si no puede usted sacar nada de ella, merece siempre la pena intercambiar las conexiones a las patillas 2 y 3 en el enchufe de la Interface 1, en caso de que hayan sido conectadas erróneamente en una primera instancia. La velocidad en baudios generada, aunque no siempre definida con precisión, es correcta dentro de la holgura tolerada por la especificación del RS232.

Como ejemplo de empleo de la interface RS232, el siguiente fragmento de código máquina transmite 255 bytes de datos por la interface serial:

```
                  ;envia 255 bytes de datos por la interface serial
                  ;
CF              rst   #8           ;establece...
31              defb  49           ;...las variables de la I/face1
216E00          ld    hl,110       ;valor para 1200 baudios
22C35C          ld    (23747),hl   ;establece veloc. en baudios
06FF            ld    b,255        ;establece contador
C5      loop:   push  bc           ;guarda contador
3E00            ld    a,DATA       ;el usuario pone la rutina...
CF              rst   #8           ;...para tomar datos del reg a
1E              defb  30           ;envia el byte contenido en a
C1              pop   bc           ;restaura contador
10F8            djnz  loop
C9              ret
```

Por muchos motivos la conexión RS232 es la más fácil de las funciones de la Interface 1 que se emplean. No hay que preocuparse por ninguna información del canal desde el código máquina, y sólo se dispone de dos llamadas —transmisión y recepción— que hay que atender.

Medios de comunicación
La puerta serial de la Interface 1 ofrece un medio de comunicación con otros dispositivos seriales y emplea un enchufe D de nueve patillas. La única desventaja de la interface serial del Spectrum es su incapacidad para manejar diferentes velocidades de transmisión en la emisión y en la recepción

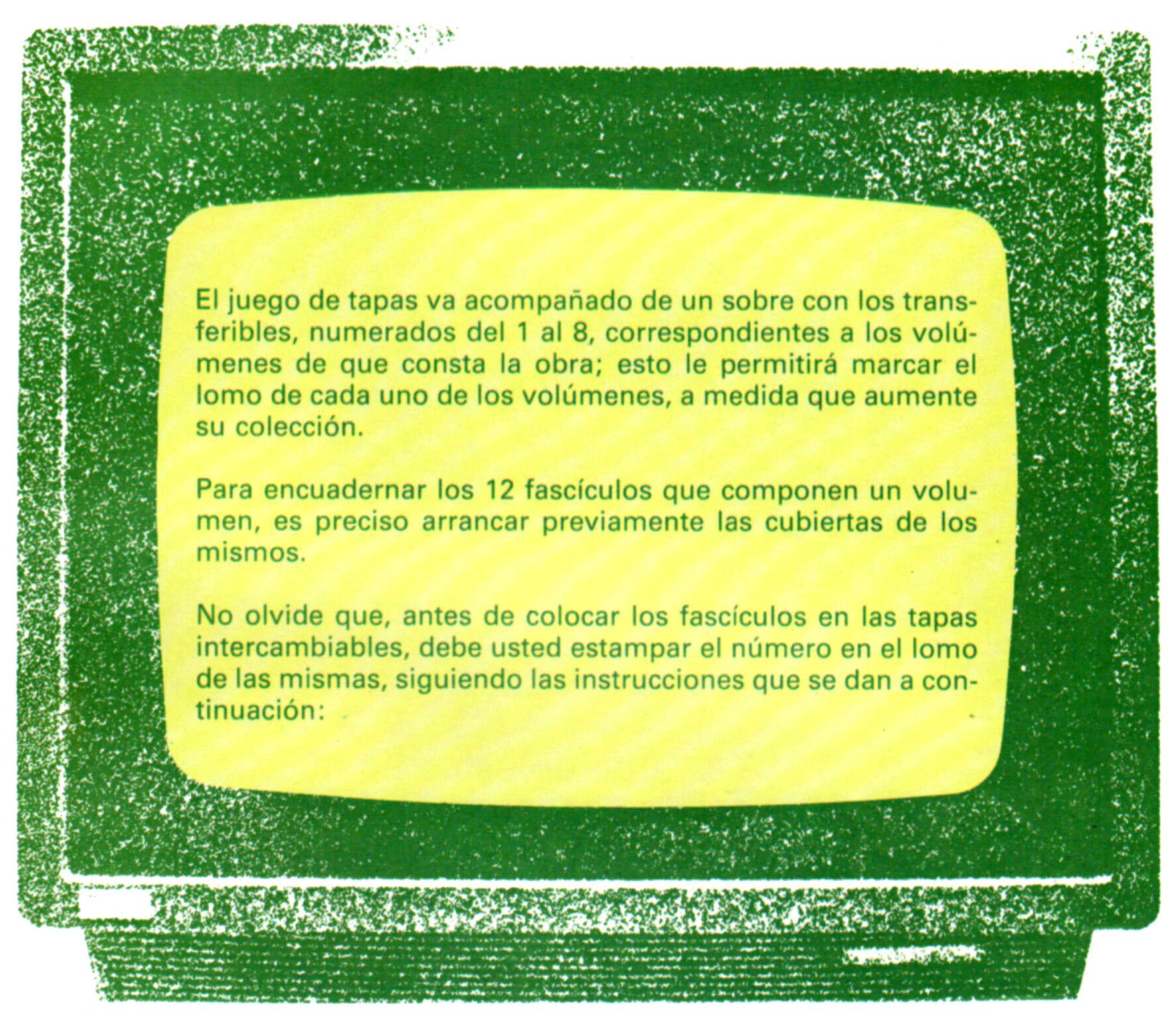
Datos básicos (III)

Continuamos el análisis del mapa de memoria del C64, con información suministrada por Commodore Business Machines

ETIQUETA	DIRECCIÓN HEXA	POSICIÓN DECIMAL	DESCRIPCIÓN
FACHO	0062-0065	98-101	Acumulador #1 c. flotante: mantisa
FACSGN	0066	102	Acumulador #1 c. flotante: signo
SGNFLG	0067	103	Puntero: constante evaluación series
BITS	0068	104	Acumulador #1 c. flotante: dígito desbordamiento
ARGEXP	0069	105	Acumulador #2 c. flotante: expon.
ARGHO	006A-006D	106-109	Acumulador #2 c. flotante: mantisa
ARGSGN	006E	110	Acumulador #2 c. flotante: signo
ARISGN	006F	111	Resultado signo comparación: Acumul. #1 frente #2
FACOV	0070	112	Acumulador #1 c. flotante: orden inferior (redondeo)
FBUFPT	0071-0072	113-114	Puntero: buffer cassette
CHRGET	0073-008A	115-138	Subrutina: tomar byte siguiente del texto BASIC
CHRGOT	0079	121	Entrada a tomar mismo byte texto BASIC
TXTPTR	007A-007B	122-123	Puntero: byte actual texto BASIC
RNDX	008B-008F	139-143	Valor orig. función RND c. flotante
STATUS	0090	144	Palabra estado E/S Kernal: ST
STKEY	0091	145	Flag: tecla STOP/tecla RVS
SVXT	0092	146	Constante temporizado para cinta
VERCK	0093	147	Flag:0=carga, 1=verificación
C3PO	0094	148	Flag:Bus serial-carac. salida en buffer
BSOUR	0095	149	Carácter en buffer bus serial
SYNO	0096	150	Número sincr. cassette
	0097	151	Área datos temporal
LDTND	0098	152	Número archivos abiertos/tabla índice archivos



Con este fascículo se han puesto a la venta las tapas correspondientes al octavo volumen



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 8, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

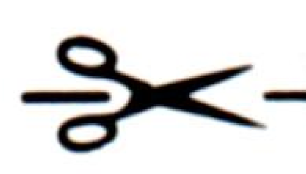
No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:

1 Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.

2 Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.

3 Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.

Cada sobre de transferibles contiene una serie completa de números, del 1 al 8, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.



PETICION DE FASCICULOS ATRASADOS

Si desea recibir algún fascículo atrasado o tapas, según las condiciones establecidas en el recuadro de la segunda página de cubierta («servicio de suscripciones y atrasados»), basta que rellene en LETRAS MAYUSCULAS este boletín y lo envíe a Editorial Delta, S.A., Aribau, 185, 1.°, 08021 Barcelona.

NOMBRE
APELLIDOS
FECHA NACIMIENTO, DIA MES AÑO
PROFESION
DOMICILIO
N.° PISO ESCALERA
CODIGO POSTAL
POBLACION
PROVINCIA

OBRA:
N.os de fascículos atrasados que desea recibir:
....
....
N.° de tapas: ....
....

Editorial Delta

9 MI COMPUTER

Editorial Delta

10 MI COMPUTER

Editorial Delta

MI COMPUTER

SE PROLONGA

Con la aparición del fascículo 96 "**MI COMPUTER**, Curso Práctico del ordenador personal, el micro y el miniordenador" llega a su fin previsto, tras casi dos años de cita obligada y agradecida con nuestros lectores.

Pero el mundo de la informática está en contínua evolución y para responder al interés demostrado por nuestros lectores y a las numerosas cartas recibidas, hemos decidido enriquecer esta obra con **DOS NUEVOS VOLUMENES,** de 12 fascículos cada uno.

A lo largo de estos 24 nuevos fascículos, los lectores encontrarán la información más completa y actual sobre los últimos avances tecnológicos en el campo de la informática.

MI COMPUTER les ha introducido en el complejo mundo de la informática; la prolongación de la obra le convertirá en un experto.



10096
9 788485 822836